Section 12
Autosend Function

The *Autosend* (*ATS*) function provides the user with the ability to send delays (Selected Controlled Departure Time or Blanket) to centers and airlines.

NOTE: In this section, inputs, outputs, and processing are not described for any *Autosend* routines that have minimal processing. Also, if a routine interface is listed in the Routine Interface section but not mentioned in the Processing section, it is because that interface is used numerous times; only major routines are mentioned in the Processing section.

Design Issues

The *ATS* process is designed to send messages to the *Router (RTR)* process using a particular addressing scheme. This method is tightly coupled to most of the modules in the *ATS* process. *ATS* utilizes Dialogue to generate the user interface screen. Characteristics of the user interface, such as button location and colors, are consistent with other Traffic Management Dialogue interfaces. The *ATS* interface runs within a Domain window and can be reduced to an icon by a button selection.

Processing Overview

ATS is a single-process function.

12.1 The ATS Process

Purpose

The *ATS* process sends selected delay information to centers and airlines. The *ATS* user interface allows the user to select Fuel Advisory Departure Trail (FADT) reports, reduce these reports into distinct sub-files, and send particular delay information. A report display area, where Selected Controlled Departure Time (SCDT) reports and their sub-files are displayed, is located beneath the user interface. Figure 12-1 shows the data flow of the *ATS* process.

Execution Control

The ATS process is initiated by selecting the ETMS Autosend icon on the Tool Manager (TMGR) bar. TMGR invokes a script named start_ats in the /etms5/com directory. The script invokes an executable file named ats from the /etms5/ats directory.

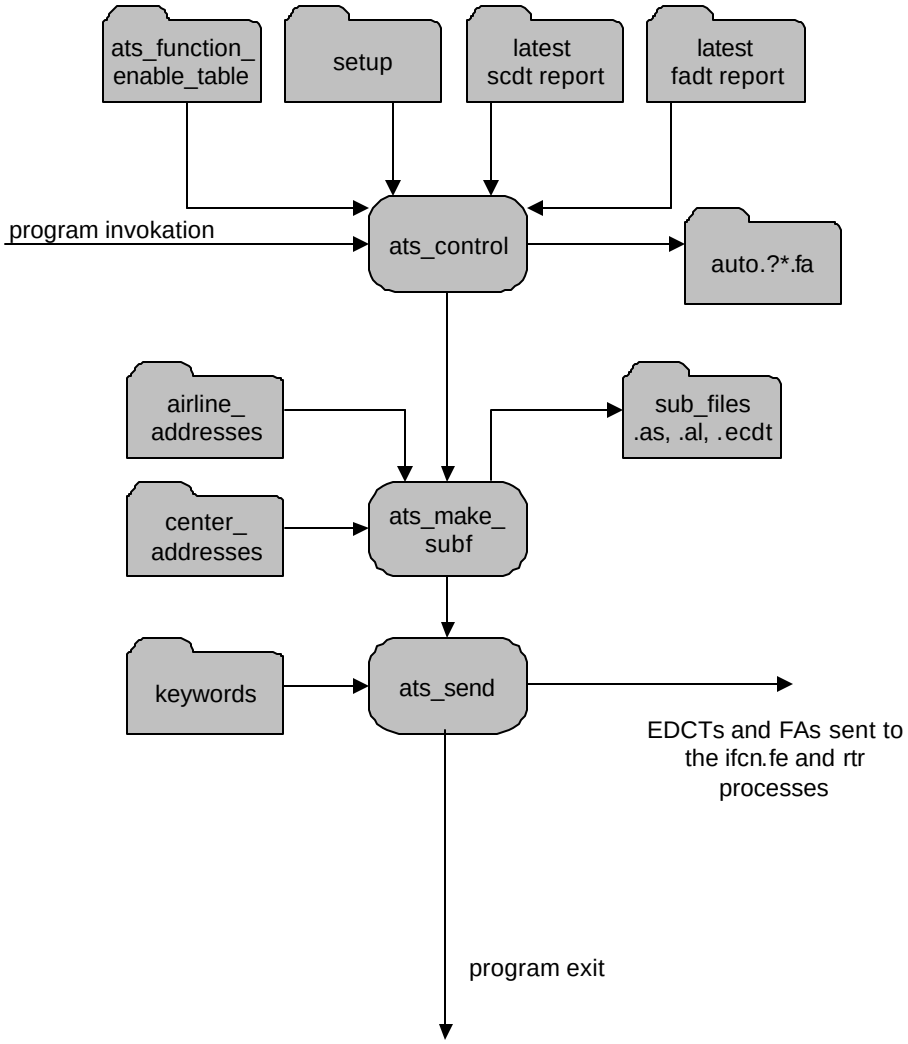

Figure 12-1. Data Flow of the Autosend Process

Processing

The *ATS* process performs the following tasks:

(1) Opens the report display area in a window beneath the user interface window.

(2) Initializes the Dialogue user interface, causing it to appear to the user.

(3) Performs initialization, including displaying the latest SCDT report, if available.

(4) Enters an event-waiting mode. In this mode, *ATS* performs the necessary processing in response to Dialogue events.

(5) Once a Dialogue quit event is detected, cleanup processing occurs, followed by the closing of the user interface and report display area windows.

Error Conditions and Handling

Descriptions of error conditions and handling are documented in each module section, where applicable.

12.2 The ats_cont Module

Purpose

This module parses the indicated field out of the supplied string buffer.

12.2.1 The ats_get_field Routine

Purpose

This routine parses the indicated field out of the supplied string buffer.

Input

string buffer

length of the buffer

field number

Output

parsed field

length of parsed field

12.2.2 The auto_extract_cont_data Routine

Purpose

This routine reads the summary section of the current SCDT report to determine if any exempt centers exist.

Input

stream handle for the current report - *ios_$id_t*

Output

exempt centers - global linked list

Processing

The *auto_extract_cont_data* routine performs the following tasks:

(1) Reads the SCDT report until the word **delay** is found.

(2) Checks the fourth word. If it equals **none**, no exempt centers exist and the routine returns. Otherwise, the centers are parsed. If a center has a delay of **one**, it is added to the exempt centers list.

The following processing is done only when the P2E version of SCDT is being used.

(3) Initializes the continuation data list.

(4) Performs the following actions for each line in the remainder of the current report:

   (a) Determines the type of data stored on the line by searching for specific keywords.

   (b) Saves the data on the line in the current record of the continuation data list. (Depending on the data, accounts for single data fields, multiple data fields, and data field qualifiers.

   (c) If a new location (airport or fix) is encountered, closes up the current record in the continuation data list and starts on next one.

(5) Saves a stack value and an acceptance rate value from the continuation data list into global variables used by other *ATS* processing.

12.2.3 The determine_line_type Routine

Purpose

This routine searches the specified line for keywords in a search key table.

Routine Interfaces

None

12.2.4 The get_data_field Routine

Purpose

This routine returns the specified field of the current SCDT line. If the field is **none**, no data is returned for the field.

Routine Interfaces

ats_get_field

12.2.5 The get_multiple_fields Routine

Purpose

This routine reads multiple fields from a data line, beginning with the specified start field. Extracted fields are stored in the specified data list.

Routine Interfaces

get_data_field

12.2.6 The init_set_table Routine

Purpose

This routine initializes the passed table for data sets. Default values are those currently being used in SCDT processes.

Routine Interfaces

None

12.2.7 The load_skip_table Routine

Purpose

This routine initializes the *Global Skip* table.

Routine Interfaces

None

12.3 The ats_control Module

Purpose

The *ats_control* module contains routines that deal with *ATS* initialization and termination. Figure 12-2 shows the data flow of the *ats_control* module.

12.3.1 The ats_open_pad Routine

Purpose

This routine opens the autosend panes within a single window for processing.

Routine Interfaces

None

12.3.2 The auto_check_for_req_files Routine

Purpose

This routine determines if files required by the *ATS* process exist.

Input

partial pathnames of files used by *ATS*

etms directory

etms directory length

Output

full pathnames and lengths

Processing

The *auto_check_for_req_files* routine checks the validity of the pathname for each of the required files. If an error occurs, *auto_display_status_message* is called to display the appropriate error message and returns to the calling procedure

Routine Interfaces

auto_display_status_message

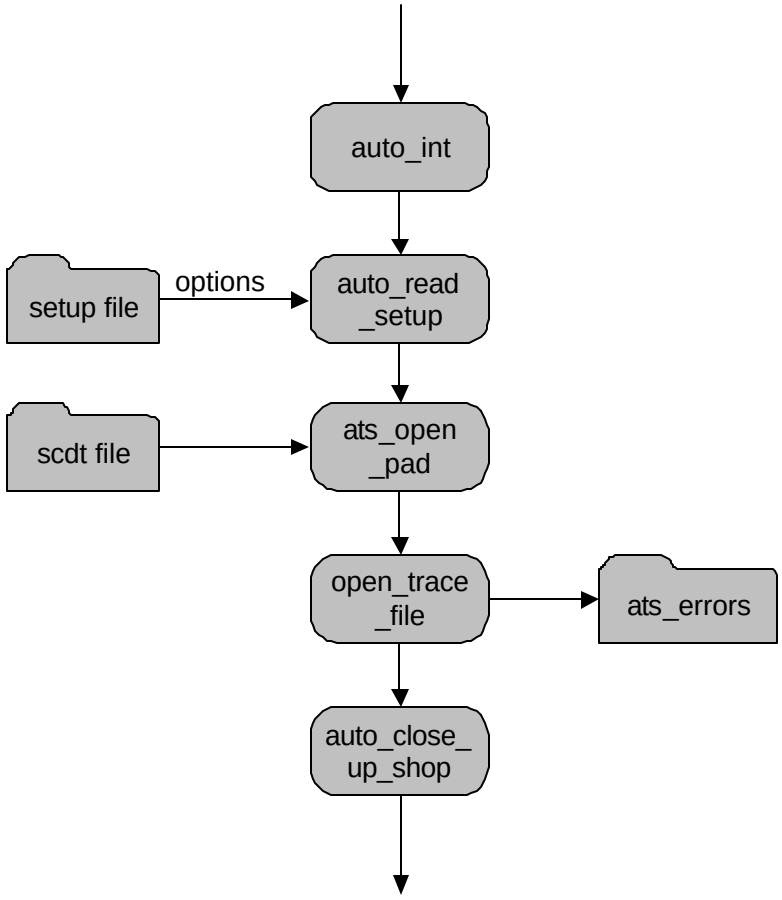

Figure 12-2. Data Flow of the ats_control Module

12.3.3 The auto_close_up_shop Routine

Purpose

This routine terminates *ATS* processing: closing all open pads, displaying a message saying *ATS* is terminating, and closing the communication channels.

Input

open *ATS* pad streams

Output

termination status message

Processing

The *auto_close_up_shop* routine performs the following tasks:

(1) Calls *auto_display_status_message* to display a termination status message, informing the user that *ATS* is in the process of terminating execution.

(2) Closes any files displayed in the report display area and the windows open for those files.

(3) Calls *load_font* to save the current font name and calls *close_com* to close the communication channel.

Routine Intefaces

load_font

auto_display_status_message

close_com

12.3.4 The auto_get_args Routine

Purpose

This routine reads arguments passed to *ATS* upon invocation and places the argument values into global variables.

Routine Interfaces

auto_display_status_message

12.3.5 The auto_init Routine

Purpose

This module performs initial *ATS* processing, including reading invocation arguments and adaptation values, displaying an initial SCDT report, and setting font characteristics.

Input

icon fonts file

latest FADT report

Output

NADIN and ARINC data

Processing

The *auto_init* routine performs the following tasks:

(1) Determines if the P2E version of SCDT is being used and if so, sets the latest file to **'/reports/?*scdt?***. If the P2E version of SCDT is not being used, sets the latest file to **'/reports/fadt?***.

(2) Validates the file by calling *auto_valid_report_name.* If the file is valid then calls *auto_display_scdt_file* to display the current SCDT file. If the current SCDT file is already displayed, calls *reset_activation_count* to reset the **make_sub_files_icon_task**.

(3) Calls *set_adaptation_values* to read the current values set in the adaptation file.

(4) Calls *auto_get_args* and *delete_auto_bak_files* to delete any existing backup files.

(5) Gets the current time and calls *auto_set_help_title, auto_read_setup* and *auto_read_send_enable* to activate startup values.

(6) Sets the position of the cursor at the approximate middle of the user interface by calling *goto_position*.

Routine Interfaces

auto_valid_report_name

auto_display_scdt_file

auto_display_status_message

set_adaptation_values

auto_get_args

delete_auto_bak_files

auto_set_help_title

auto_read_setup

auto_read_send_enable

goto_position

reset_activation_account

12.3.6 The auto_read_setup Routine

Purpose

This routine reads the setup options contained in the **/ats/config/setup** file.

Processing

The *auto_read_setup* routine performs the following tasks:

(1) Calls *auto_remove_path* to build setup name for the adaptation file.

(2) Determines if the setup options adaptation file exists. If it does, opens the file for processing. If it does not, calls *auto_display_status_message* to display a message telling the user that the adaptation file could not be opened and default options will be set.

(3) Processes the setup options adaptation file, getting the keyword from the buffer and setting the corresponding flags.

(4) Sets the default directory */reports* for advisory messages.

(5) Sets the default for extra Flow Control Advisory (FA) addresses.

Routine Interfaces

auto_remove_path

auto_display_status_message

12.3.7 The auto_read_send_enable Routine

Purpose

This routine opens the **enable option adaptation** file and processes the file contents.

Routine Interfaces

auto_remove_path

auto_display_status_message

12.3.8 The auto_set_help_title Routine

Purpose

This routine determines if the P2E version of SCDT is being used and sets the title of the help bar accordingly.

Routine Interfaces

None

12.3.9 The auto_valid_report_name Routine

Purpose

This routine determines if the passed file name is an SCDT file. In addition, if the file name is a leaf name, this routine changes it to a proper pathname.

Input

report name - of type file_list_record_t

Output

report name - of type file_list_record_t

boolean value variable

Processing

The *auto_valid_report_name* routine performs the following tasks:

(1) Reads the passed filename and checks for a /.

(2) Calls auto_remove_path to remove the passed filename.

(3) Determines the validity of the report name in the pathname. If the report name is not **fadt**, the name is invalid. Calls *auto_display_status_message* to display a message telling the user that the report name is not valid. If the report name is not found, calls *auto_display_status_message* to display a message telling the user that the airport name could not be found.

(4) If the report name is valid, encodes the report name and appends **'/reports/'** to the beginning of the name.

The following processing is done only when the P2E version of SCDT is being used.

(5) Determines the validity of the report name in the pathname. If it is **scdt** or **blan**, the name is valid.

Routine Interfaces

auto_display_status_message

auto_remove_path

12.3.10 The check_adv_path Routine

Purpose

This routine checks the specified pathname for validity and sets the adapted advisory directory.

Routine Interfaces

auto_display_status_message

12.3.11 The delete_auto_bak_files Routine

Purpose

This routine deletes all system-created **.bak** files for sub-files created during the current *ATS* session.

Routine Interfaces

None

12.3.12 The latest_file Routine

Purpose

This routine returns the name of the latest FADT file edited or created.

Routine Interfaces

None

12.3.13 The load_cutoff_value Routine

Purpose

This routine loads the specified cutoff value for *ATS* processing.

Routine Interfaces

None

12.3.14 The load_extra_fa_addr Routine

Purpose

This routine loads and validates any extra Fuel Advisory message (FA) addresses into *extra_fa_address* linked list.

Routine Interfaces

auto_validate_address

12.3.15 The load_font Routine

Purpose

This routine sets the default font that windows uses for displaying pads.

Routine Interfaces

None

12.3.16 The open_trace_file Routine

Purpose

This routine opens the **ats_trace** file where all errors occurring during processing are logged.

Routine Interfaces

auto_display_status_message

write_to_trans_pad

12.3.17 The turn_ats_to_icon Routine

Purpose

This routine changes the *ATS* user interface into an icon.

Routine Interfaces

pop_to_icon

12.4 The ats_dialog Module

Purpose

This module contains routines called by Dialog tasks.

12.4.1 The ats_min_max Routine

Purpose

This routine sets the minimum and maximum window size.

Routine Interfaces

None

12.4.2 The ats_pop_notify Routine

Purpose

This routine pops the *Notify* icon to a window.

Routine Interfaces

auto_display_status_message

12.4.3 The auto_display_gnrl_help Routine

Purpose

This routine displays the general help text to the user.

Routine Interfaces

None

12.4.4 The auto_load_into_menu Routine

Purpose

This routine loads the list of items into the menu of the passed task ID.

Routine Interfaces

None

12.4.5 The auto_process_as_popup Routine

Purpose

This routine handles the selection of a choice in the menu popup that appears when the *Send Airline File* button in the user interface is selected. If the airline delays choice is selected, another menu for selecting an airline is displayed. Otherwise, the other specified section of the airline sub-file is transmitted.

Input

global task identification

Output

selection menu

Processing

The *auto_process_as_popup* routine performs the following tasks:

(1) Determines the task ID. If FA menu is selected, calls *auto_load_into_menu* to load the FA menu options and pops up the FA menu.

(2) If the menu selection is Estimated Departure Clearance Time (EDCT), pops up the EDCT menu and loads the available airlines in the Airline menu.

(3) If more than one Air Route Traffic Control Center (ARTCC) is selected, *auto_load_into_menu* is called to set all menu choices for the ARTCC menu.

Error Conditions and Handling

If an error occurs during processing, *auto_display_status_message* is called to display a message to the effect that no airlines, ARTCCs, or hubsites are available.

Routine Interfaces

auto_load_into_menu

auto_display_status_message

12.4.6 The auto_process_can_fa_send Routine

Purpose

This routine sets the FA tasks and calls the appropriate procedure to process them.

Routine Interfaces

auto_process_send_fa_to_all

auto_process_send_fa_to_host

auto_process_send_fa_to_artccs

12.4.7 The auto_process_edct_menu_quit Routine

Purpose

This routine handles the selection of *Cancel* icon in the EDCT menu.

Routine Interfaces

auto_reset_menu

12.4.8 The auto_process_fa_menu_quit Routine

Purpose

This routine handles the selection of the *Do Not Send* icon in the FA menu.

Routine Interfaces

auto_reset_menu

12.4.9 The auto_process_make_sub_files Routine

Purpose

The *auto_process_make_sub_files* routine processes the selection of the *Make Subfiles* button.

Input

boolean value for existing sub-files

Output

displayed sub-files

Processing

The *auto_process_make_sub_files* routine performs the following tasks:

(1) Forces the title to be displayed, causing any help text that may be visible to disappear.

(2) Checks to see whether an SCDT report is currently displayed in the report display area. If a report is displayed, the routine constructs the sub-files of the report. Otherwise, the routine displays an error message.

Error Conditions and Handling

If this module is called when an SCDT report is not present in the report display area, an error message is displayed to the user.

12.4.10 The auto_process_report_popup_close Routine

Purpose

The *auto_process_report_popup_close* routine sets the enumerated value of the list reports menu task.

Routine Interfaces

None

12.4.11 The auto_process_report_selected Routine

Purpose

This *auto_process_report_selected* routine processes the user menu selection, validates all inputs, and displays the appropriate report.

Input

reports menu task identification

Output

displayed SCDT report

Processing

The *auto_process_report_selected* routine performs the following tasks:

(1) Determines and validates the files to be processed.

(2) If the file selected is **no files** (indicating that the previous report search yielded no report names), ceases report selection processing.

(3) Determines which menu selection was made, using a Dialogue system call. Uses this selection as an index into the report search file list.

(4) Validates selected report name. If name is invalid, ceases report selection processing.

(5) Determines if the selected report is already displayed in the report display area beneath the user interface. If it is, ceases report selection processing.

(6) Calls *auto_display_scdt_file* to display the selected report in the report display area.

Error Conditions and Handling

If the routine is unable to display the selected report, it calls *auto_display_status_message* to display an error message to the user.

Routine Interfaces

auto_display_status_message

auto_display_scdt_file

12.4.12 The auto_process_send_airline_delays Routine

Purpose

This routine processes the selection of the *Send Delays* button within the airline selection menu. The selected airlines have their delays sent.

Input

airline menu task identification

Output

airline delay times

Processing

The *auto_process_send_airline_delays* routine performs the following tasks:

(1) Retrieves a list of the selection status of all airlines in the airline selection menu, using a Dialogue system call.

(2) If there are no airlines in the airline selection menu, ceases processing of selected airlines.

(3) If no airlines were selected, ceases processing of selected airlines.

(4) If airlines were selected, calls *auto_send_airline_delays* to send delays for the selected airlines.

(5) If any errors occurred while sending airline information, displays an error message. Otherwise, displays a successful send status message.

(6) Deselects all airlines in the airline selection menu by calling *auto_reset_menu*.

Error Conditions and Handling

If there are airlines in the airline selection menu, airline selection processing is terminated.

If no airlines have been selected, airline selection processing is terminated.

If any error occurs during the sending of airline information, an error message is displayed to the user.

Routine Interfaces

auto_send_airline_delays

auto_reset_menu

12.4.13 The auto_process_send_edct_files Routine

Purpose

This routine sends EDCT information to the *IFCN.FE* process via ETMS to update EDCT tables. In addition, each center is sent its own subset of EDCT data. This routine is called when the *Send EDCT Files* button is selected in the *ATS* user interface.

Input

dialog task identification

Output

EDCT times

Processing

The *auto_process_send_edct_files* routine performs the following tasks:

(1) If *confirm on send* is selected in the **setup** file, sets the option to pop up a send confirmation.

(2) Determines if an EDCT sub-file is currently displayed in the report display area. If not, displays an error message and terminates sending.

(3) Gets the task value for each menu, sets the appropriate flags, and builds the files corresponding to each menu.

(4) If nothing is selected, calls *auto_display_status_message* to display a message telling the user that nothing was selected and returns to the calling procedure.

(5) Sends the entire contents of the EDCT sub-file to the *IFCN.FE* process via ETMS.

(6) Sends each center's EDCT information.

(7) Re-displays the EDCT sub-file.

(8) Reads communications adaptation values and store them in global variables.

Routine Interfaces

auto_display_status_message

auto_build_edct_center_files

auto_process_send_airline_delays

12.4.14 The auto_process_send_everything Routine

Purpose

This routine processes the selection of the *Send Everything* button on the user interface. All possible sections of all sub-files are sent.

Input

send menu task identification

Output

FA times sent to host sites, hubsite, and TMUs

EDCT time sent to hubsite, airlines, and TMUs

Processing

The *auto_process_send_everything* routine performs the following tasks:

(1) If *confirm on send* is selected in the **setup** file, sets the option to pop up a send confirmation.

(2) If sending is not available, calls *auto_display_status_message* to display a message telling the user that the communication channel is down and returns to the calling procedure.

(3) If the *Send FA* is selected, sends the FA sections of the airline sub-file.

(4) If the *Send EDCT* is selected, sends the EDCT sections of the airline sub-file.

(5) If an error occurs during processing, calls *auto_display_status_message* to display an error message to the user.

Error Conditions and Handling

If an error occurs during processing, calls *auto_display_status_message* to display an error message to the user.

Routine Interfaces

auto_display_status_message

auto_load_into_menu

auto_process_send_fa_to_all

auto_process_send_edct_files

12.4.15 The auto_process_send_fa_to_all Routine

Purpose

This routine processes the selection of the *Send* within the FA selection popup menu.

Input

FA menu task identification

Output

FA times

Processing

The *auto_process_send_fa_to_all* routine performs the following tasks:

(1) If *confirm on send* is selected in the **setup** file, sets the option to pop up a send confirmation.

(2) Gets the current menu task value. If the FA center count is less than two, *auto_display_status_message* is called to display a message saying that no centers are available for sending.

(3) Calls *goto_position* to save the **.as** pad.

(4) Calls *auto_send_fa_section_to_artccs* and *auto_send_fa_section_to_host* to send the menu choices.

(5) Resets the menu to none selected.

Error Conditions and Handling

If the communications channel is down, the routine calls *auto_display_status_message* to display a message telling the user that the communications channel is down.

Routine Interfaces

auto_display_status_message

auto_send_fa_section_at_artccs

auto_send_fa_section_to_host

goto_position

12.4.16 The auto_process_send_fa_to_artccs Routine

Purpose

This routine processes the selection of the *Send* within the FA selection popup menu.

Input

FA menu task identification

Output

FA times

Processing

The *auto_process_send_fa_to_artccs* routine performs the following tasks:

(1) If *confirm on send* is selected in the **setup** file, sets the option to pop up a send confirmation.

(2) Gets the current menu task value. If the FA center count is less than two, *auto_display_status_message* is called to display a message saying that no centers are available for sending.

(3) Calls *goto_position* to save the **.as pad** and file.

(4) Calls *auto_send_fa_section_to_artccs* to send the menu choices.

(5) Resets the menu to none selected.

Error Conditions and Handling

If the communications channel is down, calls *auto_display_status_message* to display a message telling the user that the communications channel is down.

Routine Interfaces

auto_display_status_message

auto_send_fa_section_at_artccs

goto_position

12.4.17 The auto_process_send_fa_to_host Routine

Purpose

This routine processes the selection of the *Send* within the FA selection popup menu.

Input

FA menu task identification

Output

FA times

Processing

The *auto_process_send_fa_to_host* routine performs the following tasks:

(1) If *confirm on send* is selected in the **setup** file, sets the option to pop up a send confirmation.

(2) Gets the current menu task value. If the FA center count is less than two, *auto_display_status_message* is called to display a message saying that no centers are available for sending.

(3) Calls *goto_position to s*ave the **.as** pad and file.

(4) Calls *auto_send_fa_section_to_host* to send the menu choices.

(5) Resets the menu to none selected.

Error Conditions and Handling

If the communications channel is down, the routine calls *auto_display_status_message* to display a message telling the user that the communications channel is down.

Routine Interfaces

auto_display_status_message

auto_send_fa_section_to_host

goto_position

12.4.18 The auto_process_subf_reuse_selection Routine

Purpose

The *auto_process_subf_reuse_selection* routine determines whether or not to reuse any existing sub-files.

Routine Interfaces

set_buttons_for_send

goto_position

12.4.19 The auto_quit_confirm Routine

Purpose

This routine determines whether or not to display the *quit* confirmation.

Routine Interfaces

set_buttons_for_send

goto_position

12.4.20 The auto_reset_airline_menu Routine

Purpose

This routine resets the EDCT Airline menu to *No Selections*.

Routine Interfaces

None

12.4.21 The auto_reset_menu Routine

Purpose

This routine resets the specified menu to *No Selections.*

Routine Interfaces

auto_remove_path

auto_display_status_message

12.4.22 The auto_set_cutoff_time Routine

Purpose

This routine sets the value of the cutoff time specified by the user.

Routine Interfaces

reset_activation_count

12.4.23 The auto_process_report_search Routine

Purpose

This module performs a report search, using a specified node value (if available). Results of the report search are listed in a last-time-modified order, with the most recent files at the beginning of the list.

Input

report task identification

node name

Output

listing of reports for the requested directory

Processing

The *auto_process_report_search* routine performs the following tasks:

(1) Initializes local values.

(2) Validates pathname.

(3) Sets the value entered in the user interface and date sorts the files, using a Dialogue system call.

(4) Displays the list of file names obtained from the report search in the report search popup. If there are more file names than can be displayed at one time, displays a warning message.

12.4.24 The check_all_selection Routine

Purpose

This routine determines the current task and checks all choices selected in the menu popups.

Routine Interfaces

None

12.4.25 The move_to Routine

Purpose

This routine moves the cursor to the specified place on the display.

Routine Interfaces

None

12.4.26 The move_to_window Routine

Purpose

This routine moves the cursor to the specified window on the display.

Routine Interfaces

move_to

12.4.27 The reset_activation_count Routine

Purpose

This routine resets the activation count of the specified task ID.

Routine Interfaces

None

12.4.28 The set_buttons_for_send Routine

Purpose

This routine sets the *Menu Send* buttons according to the report displayed.

Routine Interfaces

reset_activation_count

12.4.29 The window_coord Routine

Purpose

This routine sets the window coordinates (top and left position) as specified by the stream ID.

Routine Interfaces

None

12.5 The ats_gen Module

Purpose

This module contains routines called by other *ATS* modules and routines.

12.5.1 The auto_create_section_file_name Routine

Purpose

This routine creates a file name for a section of the airline sub-file, based on the name of the airline sub-file and the previous section files created.

Routine Interfaces

None

12.5.2 The auto_display_scdt_file Routine

Purpose

This routine displays the current SCDT file in the report display area. In addition, the names of the sub-files for the current SCDT report are initialized.

Routine Interfaces

auto_open_pad

auto_display_status_message

reset_activation_count

auto_make_fa-times

auto_display_scdt_name

12.5.3 The auto_display_scdt_name Routine

Purpose

This routine places the leaf portion of the current SCDT file name into the *ATS* status log.

Routine Interfaces

None

12.5.4 The auto_get_airline_addresses Routine

Purpose

This routine searches the **airline_addresses** file for the specified airline and returns the addresses for that airline in a list.

Routine Interfaces

get_word_from_buf

12.5.5 The auto_open_pad Routine

Purpose

This routine opens the report display window area at the bottom of the *ATS* user interface. In addition, this routine opens a window in the *ATS* status log and places an initialization message in the log.

Routine Interfaces

load_font

Routine Interfaces

None

12.5.6 The auto_dispose_temp_list Routine

Purpose

This routine disposes of a list of strings used commonly for temporary purposes by deallocating the memory used by each list element.

Routine Interfaces

auto_dispose_temp_list

12.5.7 The auto_display_status_message Routine

Purpose

This routine handles the selection of a choice in the menu popup that appears when the *Send Airline File* button is selected in the user interface. If the airline delays choice is selected, another menu for selecting an airline is displayed. Otherwise, the other specified section of the airline sub-file is transmitted.

Input

task_id - integer value

alert - boolean value

Output

text - character string

text_len - integer value

Processing

The *auto_display_status_message* routine performs the following tasks:

(1) If the trace option is activated, calls *write_to_trans_pad* to log errors to the trace file. If an error is displayed, returns to the calling procedure. If the pad is an icon, returns to the calling procedure.

(2) Determines if the text length is less than the maximum allowable and if so, finds the last blank space and sets an index according to the position.

(3) Determines the task ID and utilizes dialog calls to set the appropriate task values.

Error Conditions and Handling

None

12.5.8 The status_popdown Routine

Purpose

This routine pops down a status message when the *OK* button is pressed.

Routine Interfaces

None

12.6 The ats_list Module

Purpose

This module contains routines used in the construction of sub-files.

12.6.1 The ascii_time_to_sec_w_midx Routine

Purpose

This routine converts Calculated Time of Arrival (CTA) times to seconds of the day and compares the time in seconds to the current time to determine if midnight crossover has occurred.

Input

ASCII time - character string

current time - integer value

Output

seconds elapsed since midnight

Processing

The *ascii_time_to_sec_w_midx* routine performs the following tasks:

(1) If the first character in the ascii time is a letter, sets **i** to **1**; otherwise sets **i** to **0**.

(2) Sets the values of **temp[1]** and **temp[2]** to **ascii_time[1+i]** and **ascii_time[2+i]** to skip the first character if it is a letter.

(3) Calls *vfmt_$decode2* to decode the ascii time hours into integers and checks the status.

(4) Sets the values of **temp[1]** and **temp[2]** to **ascii_time[3+i]** and **ascii_time[4+i]** to skip the first character if it is a letter.

(5) Calls *vfmt_$decode2* to decode the ascii time minutes into integers and checks the status.

(6) Calculates the seconds in the ascii time and compares it to the current time in seconds.

(7) If the CTA is *greater* than the current time, midnight crossover has not occurred; the CTA is calculated by adding the seconds in the CTA hour and the seconds in the CTA minute.

(8) If the CTA is *less* than the current time, midnight crossover has occurred; the CTA is calculated by adding the seconds in the CTA hour, the seconds in the CTA minute, and the seconds in a day.

Error Conditions and Handling

If an error is encountered on the *vfmt_$encode* calls, the status message is printed.

Routine Interfaces

vfmt_$decode2

12.6.2 The calc_eta_times Routine

Purpose

This routine calculates the *estimated* time of arrival for each flight in the flight list from the flight's *calculated* time of arrival and delay.

Input

flight list

Output

Estimated Time of Arrival (ETA) for all flights in the flight list

Processing

The *calc_eta_times* routine performs the following tasks:

(1) Initializes local values.

(2) Determines that there are flights in the list.

(3) Determines if the flight is exempt. If it is exempt, goes on to the next flight in the list. If it is not exempt, calls *vfmt_$decode2* to decode the CTA to an integer value.

(4) Compares the previous time to the CTA. If the CTA is *less* than the previous time, adds **2400** to the CTA to allow for midnight crossover.

(5) Calls *inc_time* to calculate the ETA for the flight and goes on to the next flight in the list.

Error Conditions and Handling

None

Routine Interfaces

inc_time

vfmt_$decode2

12.6.3 The construct_airline_list Routine

Purpose

This routine constructs a list of individual airlines and a sub-list (for each airline) of pointers to its flights in the flight list.

Input

flight list

empty airline list

Output

airline list

Processing

The *construct_airline_list* routine performs the following tasks:

(1) Calls *create_air_sort* to create a temporary sort list, pointing to flights in the flight list in airline order.

(2) Initializes local variables.

(3) Creates a new airline record for each distinct airline, sets its fields, and constructs a sub-list pointing to its flights in the flight list.

(4) Calls *dispose_ptr_list* to dispose of the temporary sort list.

Error Conditions and Handling

None

Routine Interfaces

create_air_sort

dispose_ptr_list

12.6.4 The construct_center_list Routine

Purpose

This routine constructs a list of individual centers (ARTCCs) and a sub-list (for each center) of pointers to its flights in the flight list.

Input

flight list

empty center list

Output

center list

Processing

The *construct_center_list* routine performs the following tasks:

(1) Creates a temporary sort list pointing to flights in the flight list in center order.

(2) Calls *create_center_sort* to create a new center record for each distinct center, sets its fields, and constructs a sub-list pointing to its flights in the flight list.

(3) Calls *dispose_ptr_list* to dispose of the temporary sort list.

Error Conditions and Handling

None

Routine Interfaces

create_center_sort

dispose_ptr_list

12.6.5 The construct_delay_list Routine

Purpose

This routine constructs a list of average delays for each 15-minute interval from the flight list. All adjacent time intervals having the same average delay are grouped as one interval, and the first interval always starts at the beginning of the first hour for the current report.

Input

flight list

empty delay list

Output

delay list

Processing

The *construct_delay_list* routine performs the following tasks:

(1) Calculates correct ETA times for each flight in flight list, using each flight's CTA time and delay to account for midnight crossover, as necessary.

(2) Constructs a temporary sort list, pointing to flights in the flight list in ETA order.

(3) Starting with the first non-exempt flight at the beginning of the hour, calculates average delays for flights falling in 15-minute intervals and stores this information in the delay list.

(4) Checks through the entire delay list, combining adjacent intervals with identical average delays.

(5) Disposes of the temporary sort list.

Routine Interfaces

calc_eta_times

inc_time

create_eta_sort

dispose_ptr_list

12.6.6 The create_air_sort Routine

Purpose

This routine creates a list of pointers to the flights in the flight list in airline order.

Input

flight list

Output

airline sorted flight list

Processing

The *create_air_sort* routine performs the following tasks:

(1) Initializes local variables.

(2) Initializes the flight pointer list.

(3) Sets up a pointer to the first flight in the flight list and puts it in the sorted pointer list.

(4) Sets up a pointer to the next flight in the flight list, checks the flight identifier against the other flights in the sorted list, and places the pointer to the flight in the appropriate position in the sorted list.

Error Conditions and Handling

None

Routine Interfaces

None

12.6.7 The create_air_sort_by_cta Routine

Purpose

This routine creates a list of pointers to the flights in the flight list in airline order by CTA.

Input

flight list - of type flight_rec

Output

airline sorted flight list - of type flight_rec

Processing

The *create_air_sort_by_cta* routine performs the following tasks:

(1) Calls *cal_$decode_clock* to get the current clock time.

(2) Calculates the current clock time in seconds.

(3) Initializes local variables.

(4) Initializes the flight pointer list.

(5) Sets up a pointer to the first flight in the flight list, calls *ascii_time_to_sec_w_midx* to convert the CTA to seconds, and puts it in the sorted pointer list.

(6) If the third character in the **ident** field is a letter, sets **current_airline_len** to **3**.
If it is not a letter, sets **current_airline_len** to **2**.

(7) Calls *strncpy* to copy the current airline to the **ident** field of the flight record.

(8) Puts the pointer to the flight record in the sorted list. If it is not the first flight, compares the **current_airline_len** and **current_airline** to the other sorted list entries. If both records are for the same airline**,** compares the CTAs and places the new pointer in the sorted list in the appropriate position.

Error Conditions and Handling

None

Routine Interfaces

cal_$decode_clock

strncpy

ascii_time_to_sec_w_midx

12.6.8 The create_center_sort Routine

Purpose

This routine creates a list of pointers to the flights in the flight list sorted by departure center.

Input

flight list - of type flight_rec

Output

flight list sorted by center - of type flight_rec

Processing

The *create_center_sort* routine performs the following tasks:

(1) Initializes local variables.

(2) Initializes the flight pointer list.

(3) Sets up a pointer to the first flight in the flight list and puts it in the sorted pointer list.

(4) Sets up a pointer to the next flight in the flight list, checks the flight identifier against the other flights in the sorted list, and places the pointer to the flight in the appropriate position in the sorted list.

Error Conditions and Handling

None

Routine Interfaces

None

12.6.9 The create_eta_sort Routine

Purpose

This routine creates a list of pointers to the flights in the flight list sorted by estimated time of arrival

Input

flight list - of type flight_rec

Output

flight list sorted by ETA - of type flight_rec

Processing

The *create_eta_sort* routine performs the following tasks:

(1) Initializes local variables.

(2) Initializes the flight pointer list.

(3) Sets up a pointer to the first flight in the flight list and puts it in the sorted pointer list.

(4) Sets up a pointer to the next flight in the flight list, checks the flight identifier against the other flights in the sorted list, and places the pointer to the flight in the appropriate position in the sorted list.

Error Conditions and Handling

None

Routine Interfaces

None

12.6.10 The create_flight_list Routine

Purpose

This routine reads all the flight records from the current SCDT report and stores the important fields in a flight list sorted by flight identification.

Input

stream handle for the current report

flight records from the current report - of type flight_rec

empty flight list - of type flight_rec

Output

flight list - of type flight_rec

Processing

The *create_flight_list* routine performs the following tasks:

(1) Initializes local variables.

(2) Calls *ios_$get* to read a record from the SCDT or Blanket file and attempts to read the first word from the record, using the *get_field_from_rpt* routine. If the record can be read and a word can be extracted, the record is a flight.

(3) Initializes the flight list record.

(4) Determines if the P2E version of SCDT is being used.

(5) Copies the word extracted from the read record into the **ident** field of the current flight record.

(6) Extracts the departure airport field from the read record.

(7) Extracts the **ptime** field from the read record. If the first character is a letter, determines if it is **A** or **a** (signifying an active flight). If the flight is active, sets the *active* flag for the flight and copies the time of the **ptime** field to the current flight record. If the flight is not active, the **ptime** field is copied to the current flight record.

(8) Extracts the **fix** field from the read record.

(9) Extracts the **cta** field from the read record. If the first character is non-numeric, copies only the numeric portion of the **cta** field to the current flight record. Otherwise, copies the whole **cta** field.

(10) Extracts the **delay** field from the read record. Calls *vfmt_$decode* to decode the **delay** field.

(11) Extracts the **ctime** field from the read record. If the first character is a **#**, the flight is active and the *active* flag is set.

(12) Attempts to extract another field from the read record. If successful and the word extracted is *****, the *exempt* flag is set.

(13) For each report line containing flight information, creates a new flight list record, records the flight information, and adds the flight list record to the flight list.

(14) Calls *filter_active_stack_ga_flights* routine to filter all active, STACK, and General Aviation (GA) flights from the flight list.

The following processing is done only when the P2E version of SCDT is being used.

(15) Copies the word extracted from the read record into the **ident** field of the current flight record.

(16) Extracts other words from the record and puts them into the current flight record.

(17) Extracts the **ptime** field from the record. If the first character is a letter, determines if it is **A** or **a** (signifying an active flight). If the flight is active, sets the *active* flag for the flight and copies the time of the **ptime** field to the current flight record. If the flight is not active, the **ptime** field is copied to the current flight record.

(18) Extracts the **ctime** field from the record. If the first character is a **#**, the flight is active and the *active* flag is set.

(19) Extracts the **eta** field from the record.

(20) Extracts the **cta** field from the record. If the first character is non-numeric, copies only the numeric portion of the **cta** field to the current flight record. Otherwise, copies the whole **cta** field.

(21) Extracts the **delay** field from the record. Calls *vfmt_$decode* to decode the **delay** field.

(22) Attempts to extract another field from the record. If successful and the word extracted is *****, the *exempt* flag is set.

(23) Calls *filter_international_centers* routine to filter all flights for an international center.

Error Conditions and Handling

None

Routine Interfaces

filter_active_stack_ga_flights

filter_international_centers

get_field_from_rpt

ios_$get

vfmt_$decode

12.6.11 The dispose_airline_list Routine

Purpose

This routine disposes of the airline list by deallocating the memory used by each element in the list.

Input

airline list

Output

None

Processing

The *dispose_airline_list* routine performs the following tasks:

(1) If **list** is not equal to **nil**, calls *dispose_airline_list* to deallocate the pointers.

(2) Calls *dispose* to deallocate the list.

Error Conditions and Handling

None

Routine Interfaces

dispose_airline_list

dispose

12.6.12 The dispose_center_list Routine

Purpose

This routine disposes of a center list by deallocating the memory used by each list element.

Input

center list - of type center_list_rec

Output

disposed memory previously allocated for the center linked list.

Processing

The *dispose_center_list* routine performs the following tasks:

(1) If **list** is not equal to **nil**, calls *dispose_center_list* to deallocate the pointers.

(2) Calls *dispose* to deallocate the list.

Error Conditions and Handling

None

Routine Interfaces

dispose_center_list

dispose

12.6.13 The dispose_delay_list Routine

Purpose

This routine disposes of the specified delay list by deallocating the memory used by the list elements.

Input

delay list

Output

disposed memory

Processing

The *dispose_delay_list* routine performs the following tasks:

(1) If **list** is not equal to **nil**, calls *dispose_delay_list* to deallocate the pointers.

(2) Calls *dispose* to deallocate the list.

Error Conditions and Handling

None

Routine Interfaces

display_delay_list

display

12.6.14 The dispose_flight_list Routine

Purpose

This routine disposes of the specified flight list by deallocating the memory used by the list elements.

Input

flight list

Output

disposed memory

Processing

The *dispose_flight_list* routine performs the following tasks:

(1) If **list** is not equal to **nil**, calls *dispose_flight_list* to deallocate the pointers.

(2) Calls *dispose* to deallocate the list.

Error Conditions and Handling

None

Routine Interfaces

dispose_flight_list

dispose

12.6.15 The dispose_ptr_list Routine

Purpose

This routine disposes of a list whose elements are pointers to flight records by deallocating the memory used by each list element.

Input

flight list

Output

disposed memory

Processing

The *dispose_ptr_list* routine performs the following tasks:

(1) If **list** is not equal to **nil**, calls *dispose_ptr_list* to deallocate the pointers.

(2) Calls *dispose* to deallocate the list.

Error Conditions and Handling

None

Routine Interfaces

dispose_ptr_list

dispose

12.6.16 The filter_international_centers Routine

Purpose

This routine disposes of all flights from the flight list with an international center name.

Input

flight list

Output

filtered flight list - of type flight_list_rec

error messages - character string

Processing

The *filter_international_centers* routine performs the following tasks:

(1) Gets the pathname for the **international_centers** file.

(2) Opens the **international_centers** file.

(3) If the file cannot be opened, calls *auto_display_status_message* to display an error.

(4) Checks each international center name in the **international_centers** file against the flight list that was passed into the routine. If the center name is in the flight list, removes it from the list.

Error Conditions and Handling

If the **international_centers** file cannot be opened, call *auto_display_status_message* to display an error.

Routine Interfaces

read_rec_from_file

get_word_from_bufa

auto_display_status_message

12.6.17 The filter_active_stack_ga_flights Routine

Purpose

This routine disposes of all active flights from the flight list.

Input

flight list

Output

filtered flight list

Processing

The *filter_active_stack_ga_flights* routine performs the following tasks:

(1) Calls *strncmp* to check whether the flight list contains any active, General Aviation (GA), or STACK flights.

(2) If an active, GA, or STACK flight is found, removes it from the flight list.

Error Conditions and Handling

None

Routine Interfaces

strncmp

12.6.18 The get_field_from_rpt Routine

Purpose

This routine returns the next field in the specified buffer, using the specified pointer as a starting search position.

Input

buf - character string

buf_len - integer value

pointer - integer value

Output

pointer - integer value

word - character string

length - integer value

Processing

The *get_field_from_rpt* routine performs the following tasks:

(1) Increments the pointer when it is not at the end of the buffer and not in the specified character set.

(2) If the pointer is in the specified character set, increments **length**; if **length** is less than **9**, sets the word length to the pointer and increments the pointer.

Error Conditions and Handling

None

Routine Interfaces

None

12.6.19 The inc_time Routine

Purpose

This routine returns the value of time plus interval.

Input

time

interval

Output

time

Processing

The *inc_time* routine performs the following tasks:

(1) Calculates **hour**.

(2) Calculates **min**.

(3) Determines if the interval is negative and adjusts **hour** and **min** for the previous hour.

(4) Adjusts **hour** if the interval is from the previous day.

(5) Calculates the increment time.

Error Conditions and Handling

None

Routine Interfaces

None

12.7 The ats_make_subf Module

Purpose

This module contains routines used to break down the SCDT report into three sub-files. Figure 12-3 shows the data flow of the *ats_make_subf* module.

12.7.1 The add_centers_to_menu_array Routine

Purpose

This routine loads an array with non-exempt centers used for the *FM* section.

Input

list of centers

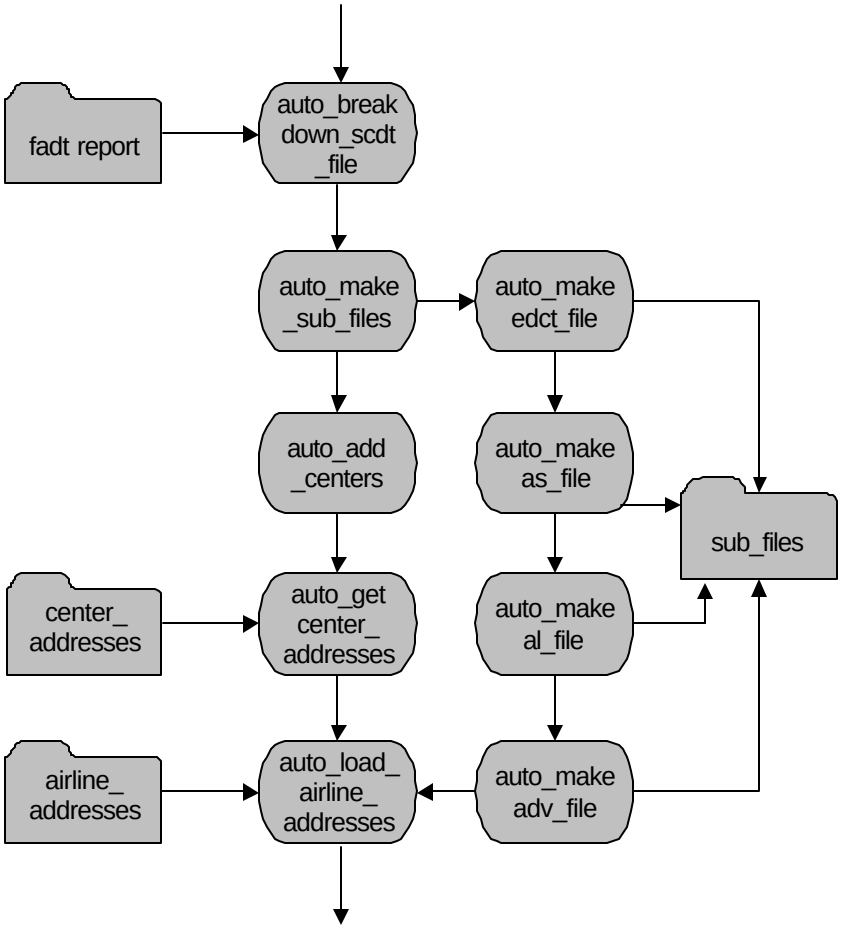
fadt report
auto_break down_scdt _file
auto_make _sub_files
auto_make edct_file
auto_add _centers
auto_make as_file
sub_files
center_ addresses
auto_get center_ addresses
auto_make al_file
airline_ addresses
auto_load_ airline_ addresses
auto_make adv_file

Figure 12-3. Data Flow of the ats_make_subf Module

Output

array of non-exempt centers

Processing

The *add_centers_to_menu_array* routine performs the following tasks:

(1) Initializes the menu arrays for the **all centers** menu value.

(2) When the center list is not empty, reads each address in the list and determines if it is exempt. If the address is not exempt, adds it to the list of non-exempt centers.

Error Conditions and Handling

None

Routine Interfaces

get_word_from_buf

12.7.2 The add_1 Routine

Purpose

This routine increments the time hour value by one.

Input

hour value - two characters

Output

incremented hour value - two characters

Processing

The *add_1* routine performs the following tasks:

(1) If the second character of a two-character number is **9**, increments the first character and sets the second character to **0**. If the second character is not **9**, increments only the second character.

(2) If the number is **23**, sets the number to **00**.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.3 The ats_read_in_preface_file Routine

Purpose

This routine reads lines from the specified preface file and stores that data in the specified preface list.

Input

file_name - character string

file_name_len - integer value

preface_list - linked list of records of type temp_rec

Output

preface_list - linked list of records of type temp_rec

Processing

The *ats_read_in_preface_file* routine performs the following tasks:

(1) Initializes local variables.

(2) Attempts to open the preface file and check the status. If a problem is encountered on opening the preface, calls *auto_display_status_message* to display an error message.

(3) Attempts to read the preface file.

Error Conditions and Handling

Attempts to open the preface file to check the status. If a problem is encountered on opening the preface, the routine calls *auto_display_status_message* to display an error message.

Routine Interfaces

strncpy

auto_display_status_message

12.7.4 The auto_add_centers Routine

Purpose

This routine adds a center to a temporary linked list.

Input

None

Output

None

Processing

The *auto_add_centers* routine performs the following task:

Adds the new center passed in to the temporary center list.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.5 The auto_break_down_scdt_file Routine

Purpose

This routine validates the format of the current SCDT report and extracts the destination airport, flight information, and continuation information from the report.

Input

SDCT report

Output

status messages

Processing

The *auto_break_down_scdt_file* routine performs the following tasks:

(1) Opens the current SCDT. If unable to open, determines if the P2E version of SCDT is being used, displays an error message, and ends data extraction processing.

(2) Calls *auto_process_flights* to extract flight data from the current report.

(3) Calls *auto_extract_cont_data* to extract continuation data from the summary section of the current report.

(4) Closes the current report.

The following processing is done only when the P2E version of SCDT is being used.

(5) Calls *auto_validate_sedt_report* to validate the format of the report. If it is not valid, closes the file, sets a global status variable to a bad status, and ends data extraction processing.

Error Conditions and Handling

If the current SCDT report cannot be opened for reading, *auto_display_status_message* is called to display an error message, and data extraction processing is terminated.

If the format of the current report is invalid, a global status variable is set to a bad status and data extraction processing is terminated.

Routine Interfaces

auto_process_flights

auto_display_status_message

auto_validate_scdt_report

auto_extract_cont_data

12.7.6 The auto_check_dup_center Routine

Purpose

This routine deletes duplicate centers from a linked list.

Input

center list - linked list of records of type center_list_rec

Output

center list - linked with duplicates eliminated

Processing

The *auto_check_dup_center* routine performs the following task:

When there are centers in the list, checks for duplicates and deletes them.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.7 The auto_compare_center_lists Routine

Purpose

This routine compares two center lists to determine if there are duplicates in the list.

Input

center lists - linked list of records of type center_list_rec

Output

deleted center list - linked list of type cent_list_rec

Processing

The *auto_compare_center_lists* routine performs the following tasks:

(1) Sets the first center list passed into the routine to the *base* list and the second list passed in to the *compare* list.

(2) While there are centers in the *compare* list, compares them to the centers in the *base* list and deletes the duplicate centers from the second list.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.8 The auto_create_non_exempt_center_list Routine

Purpose

This routine creates a list of non-exempt centers to use for the FA computer section.

Input

non-exempt center list

Output

non-exempt center list

Processing

The *auto_create_non_exempt_center_list* routine performs the following tasks:

(1) Determines if a new list needs to be created or if the FA center list should be used. If the FA center list is used, sets the non-exempt centers list to the FA center list and returns.

(2) Creates the non-exempt centers list.

Error Conditions and Handling

None

Routine Interfaces

strncmp

auto_add_center

auto_excluded_centers

auto_check_dup_center

12.7.9 The auto_excluded_centers Routine

Purpose

This routine checks all the center lists to verify that each excluded center is included in all lists if all of the fixes are excluded.

Input

None

Output

non-exempt center list

Processing

The *auto_excluded_centers* routine performs the following tasks:

(1) If there is only one list, the centers in the list are the exempt centers.

(2) Skips the first list if it is the airport list.

(3) Creates an initial exempt centers list and deletes center names from it as no matches are found.

(4) Checks all the departure centers in the rest of the lists, deletes any centers not in all of the E centers list, and tests the center to ensure it is not in both the FA center list and the exempt center list.

(5) Calls *auto_compare_center_lists* to compare the exempt center list to the non-exempt center list.

(6) Calls *auto_dispose_temp_list* to get rid of the exempt centers list.

Error Conditions and Handling

None

Routine Interfaces

strncmp

auto_add_center

auto_compare_center_lists

auto_dispose_temp_list

12.7.10 The auto_get_center_address Routine

Purpose

This routine returns center name, center address, computer code, and a flag indicating if the center has no information.

Input

fa_list_ptr - linked list

Output

center_name - character string

center_name_len - integer value

center_address - character string

center_address_len - integer value

computer_code - characters string

computer_code_len - integer value

no_more_centers - boolean value

Processing

The *auto_get_center_address* routine performs the following tasks:

(1) Initializes output lengths to **0**.

(2) Determines if there are any more centers and returns if there are none.

(3) Calls *get_word_from_buf* to extract the center data.

Error Conditions and Handling

None

Routine Interfaces

get_word_from_buf

12.7.11 The auto_get_char_for_heading Routine

Purpose

This routine returns the next character to be put in a section header in the airline sub-file.

Input

heading character

Output

heading character

Processing

The *auto_get_character_for_heading* routine performs the following task:

Gets the next heading character and assigns it to the correct character for the section header.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.12 The auto_load_airline_addresses Routine

Purpose

This routine loads the airline addresses into the airline sub-file menu.

Input

airline sub-file, airline addresses

Output

None

Processing

The *auto_load_airline_addresses* routine performs the following tasks:

(1) Opens the airline sub-file. Displays an error message and returns if a bad status is returned during the open.

(2) Reads the file until **=1=** or a bad status is returned. Displays an error message, closes the file, and returns when the status is bad.

(3) Initializes the airline count and sets the default menu choice to the **all airlines** address.

(4) Loads the address from each menu choice into the airline menu.

(5) Closes the airline sub-file.

Error Conditions and Handling

None

Routine Interfaces

strncmp

get_word_from_buf

auto_display_status_message

12.7.13 The auto_load_centers_for_edct_file Routine

Purpose

This routine loads the centers into the send menu for the EDCT file.

Input

None

Output

None

Processing

The *auto_load_centers_for_edct_file* routine performs the following tasks:

(1) Opens the EDCT file. Displays an error message and ends processing if a bad status is returned.

(2) Sets **hub_menu_array** and **artcc_menu_array** first choices to **all centers**.

(3) Loads the international_centers list.

(4) Reads the EDCT file. Displays an error message if unable to read the file and not at the end of the file.

(5) Searches for center names, compares them against the hub_center_list and the artcc_center_list, and places the center in the appropriate menu.

(6) Closes the EDCT file.

Error Conditions and Handling

Displays an error message if a bad status is returned.

Routine Interfaces

strncmp

get_word_from_bufa

auto_display_status_message

12.7.14 The auto_make_adv_file Routine

Purpose

This routine constructs and displays the advisory sub-file.

Input

None

Output

None

Processing

The *auto_make_adv_file* routine performs the following tasks:

(1) Constructs the advisory sub-file name derived from the current SCDT report name.

(2) Opens the **.adv** file for writing. Displays an error message and stops processing the file if bad status is returned.

(3) Writes the first line to the **.adv**. file

(4) Writes the arrival airport, date, and delay program start time in the second line of the **.adzy** file.

(5) Writes the average delay times in the **.adv** file if the setup option is **true**.

(6) Closes the advisory file and displays a status message saying that the advisory is being created.

Error Conditions and Handling

If an error occurs when creating the file, an error is placed in the *ATS* status log and file construction is terminated.

Routine Interfaces

strncpy

auto_remove_path

auto_open_pad

auto_write_fa_time_line

auto_display_status_message

12.7.15 The auto_make_al_file Routine

Purpose

This routine creates and displays the airline sub-file of the current SCDT report, using data extracted from the report.

Input

data extracted from the SCDT

Output

airline sub-file

error message

Processing

The *auto_make_al_file* routine performs the following tasks:

(1) Removes the airline sub-file if it is currently displayed in the report display area.

(2) Opens the **.as** file for writing. Displays an error message and returns when the status is bad.

(3) Calls *auto_make_as_arrival_time_section* and *auto_make_as_fa_computer_section* to write the data to the **.as** file.

(4) Closes and saves the airline sub-file.

(5) Displays the airline sub-file in the leftmost pane of the report display area.

Error Conditions and Handling

If an error occurs when creating the new airline sub-file, an error is placed in the *ATS* status log and airline sub-file construction is terminated.

Routine Interfaces

strncpy

auto_open_pad

auto_display_status_message

auto_make_as_airline_section

12.7.16 The auto_make_as_airline_section Routine

Purpose

This routine constructs the airline section of the airline sub-file. In addition, a global menu of all airlines is created for the Airline Send Selection menu in the user interface.

Input

airline_preface file

airline_addresses file

Output

EDCT times - grouped by airline

Processing

The *auto_make_as_airline_section* routine performs the following tasks:

(1) Initializes the counter for number of airlines.

(2) Opens the **airline_addresses** file for reading. If unable to open this file, displays an error message and ends airline section creation processing.

(3) Reads in the preface information to be placed prior to each airline section.

(4) Makes **all airlines** the first entry in the Dialogue selection menu.

(5) Opens the current FADT report. Displays an error message and ends processing on the report if the file could not be opened

(6) Reads each line of the FADT report until **airline** is found or a bad status is returned. Displays an error message, closes the file, and ends report processing if a bad status is returned.

(7) Reads the FADT line. If **airline** is found, disposes of the airline addresses list and gets the airline addresses from the **airline_addresses** file.

(8) If there are addresses in the airline addresses list and an address is *not* exempt, puts the heading out for the airline.

(9) Gets the **aircraft ID**, the **departure center**, the **ptime**, the **etime**, and the **ctime** from the FADT report; calculates the **delay**; and writes all items to the airline sub-file.

(10) Closes the FADT report.

(11) Displays an error message for the airlines for which *no* addresses were found.

(12) Disposes of the *preface_list*.

(13) Closes the **airline_addresses** file

The following processing is done only when the P2E version of SCDT is being used.

(14) Determines if the airline is in the **airline_addresses** file. If it is, puts the heading out for the airline, puts out a line for each flight for the airline, and disposes of the airline_addresses list.

Error Conditions and Handling

If the **airline_addresses** file *cannot* be opened, an error message is placed in the *ATS* status log and airline section construction is terminated.

If an airline does *not* have addresses in the **airline_addresses** file, an error message is placed in the *ATS* status log, the airline is excluded from the airline section, and processing continues with the next airline.

If a bad status is returned while reading the FADT report, processing ends and an error message displays.

Routine Interfaces

strncmp

strncpy

cal_delay

write_heading

auto_write_flight_rec

get_word_from_buf

ts_read_in_preface_file

auto_dispose_temp_list

auto_get_char_for_heading

auto_display_status_message

auto_get_airline_addresses

12.7.17 The auto_make_as_arrival_time_section Routine

Purpose

This routine writes the arrival time sections in the airline sub-file.

Input

contents of the FA section preface file

Output

average delays along with start and stop interval times

Processing

The *auto_make_as_arrival_time_section* routine performs the following tasks:

(1) Reads in the contents of the FA section preface file and places this information above each FA time section.

(2) Prior to writing data for each FA time section, places a header containing the string **#?#** (where **?** is a section number starting from **1)**.

(3) Writes average delays along with start and stop interval times. Only a certain amount of information is allowed per line, and only a certain number of lines are allowed per FA time section.

(4) Disposes of the temporary list containing preface information.

Error Conditions and Handling

None

Routine Interfaces

auto_get_char_for_heading

auto_write_fa_time_line

ats_read_in_preface_file

auto_dispose_temp_list

12.7.18 The auto_make_as_fa_computer_section Routine

Purpose

This routine constructs the Host FA sections of the **.as** sub-file.

Input

FA center list - list of non-exempt ARTCCs

FA times and delays

Output

Host FA section of the .as sub-file

Processing

The *auto_make_as_fa_computer_section* routine performs the following tasks:

(1) Determines the number of centers to be included in the FM sections.

(2) Determines the number of delay intervals.

(3) Calculates the number of FM headings required from number of centers and number of delay intervals.

(4) Updates the FM sequence number for the current node using the required number of FM headings. In addition, returns the sequence number constant that is also present in the sequence file.

(5) Performs the following actions for each center in the FM section:

  (a) Writes a new heading for each line of FA times. The header contains an incremental sequence number, the address for the center, and the **+?+** string (where **?** is a section number starting with **1** for each center).

  (b) Writes a line of FA times. Each center has multiple sections, each of which has only one line of FA times.

Error Conditions and Handling

None

Routine Interfaces

strncpy

auto_get_center_address

auto_get_char_for_heading

auto_write_fa_time_line

auto_create_non_exempt_list

auto_update_fa_sequence_number

12.7.19 The auto_make_as_file Routine

Purpose

This routine creates and displays the airline sub-file of the current SCDT report, using data extracted from the report.

Input

data extracted from the SCDT report.

Output

airline sub-file

error message

Processing

The *auto_make_as_file* routine performs the following tasks:

(1) Removes the airline sub-file if it is currently displayed in the report display area.

(2) Opens the **.as** file for writing. Displays an error message and returns when the status is bad.

(3) Calls *auto_make_as_arrival_time_section* and *auto_make_as_fa_computer_section* to write the data to the **.as** file.

(4) Closes and saves the airline sub-file.

(5) Displays the airline sub-file in the leftmost pane of the report display area.

Error Conditions and Handling

If an error occurs when creating the new airline sub-file, an error is placed in the *ATS* status log and airline sub-file construction is terminated.

Routine Interfaces

auto_make_as_arrival_time_section

auto_make_as_fa_computer_section

auto_open_pad

auto_display_status_message

auto_make_as_airline_section

strncpy

12.7.20 The auto_make_edct_file Routine

Purpose

This routine constructs and displays the EDCT sub-file using information from the FADT file.

Input

FADT file

Output

.edct file sub-file

Processing

The *auto_make_edct_file* routine performs the following tasks:

(1) Removes the EDCT sub-file from the report display area if it is currently displayed.

(2) Opens the EDCT sub-file for writing. If unable to open this file, displays an error message and ends EDCT sub-file construction.

(3) Places the FADT header line in the EDCT file. This line contains time information, stack value, and acceptance rate and is included for compatibility reasons.

(4) Opens the FADT file. If unable to open this file to read the B8 section, displays an error message, closes the file, and ends processing of the FADT file.

(5) Gets the FADT line. Displays an error message, closes the FADT and the EDCT files, and ends processing of these files if a bad status is returned.

(6) Displays an error message, closes the FADT and the EDCT files, and ends processing of these files if the first line of the FADT file is not **fadt**.

(7) Writes the **fadt** line to the *EDCT* file.

(8) Reads the FADT file and searches for the B6 report. Displays an error message, closes the EDCT and FADT files, and ends processing of the files if a bad status is returned.

(9) Reads the B6 report and writes the data to the EDCT file until the B8 report is found or the end-of-file marker is reached. Displays an error message, closes the EDCT and FADT files, and ends processing of the files if the FADT file cannot be read.

(10) Displays an error message if the B6 report is not found.

(11) Closes the FADT file.

The following processing is done only when the P2E version of SCDT is being used.

(12) Initializes **first_hour** and **last_hour**.

(13) While there are delays in the delay list, copies the delay start time to **first_hour** and the stop time to **last_hour**.

(14) Searches for any **ete** other than the default value and sets the last one found.

(15) Writes the **fadt** line to the *EDCT* file.

(16) When there are centers in the center list, checks each center's validity by determining if the center name is in the **center_addresses** file. If the center is valid, places a header line for the center (including center name), followed by flights for the center. If the center is not valid, displays an error message and continues to the next center.

(17) Closes and saves the EDCT sub-file.

(18) Displays the EDCT sub-file in the middle pane of the report display area.

Error Conditions and Handling

If unable to open the EDCT sub-file for writing, an error message is placed in the *ATS* status log and the EDCT sub-file construction is terminated.

If a center with flights is *not* listed in the **center_addresses** file, an error message is displayed in the *ATS* status log, the center's flights are *not* placed in the EDCT sub-file, and processing continues with the next center.

If the FADT file cannot be opened to read the B8 section, the routine displays an error message, closes the file, and ends processing of the FADT file.

If the first line of the FADT file is not **fadt**, the routine displays an error message, closes the FADT and the EDCT files, and ends the processing of these files.

If a bad status is returned, the routine reads the FADT file and searches for the B6 report, displays an error message, closes the EDCT and FADT files, and ends the processing of the files.

If the B6 report is not found, displays an error message.

Routine Interfaces

add_1

strncpy

strncmp

cal_delay

get_word_from_buf

open_ios_f2_write

auto_valid_center

auto_write_flight_rec

auto_display_status_message

12.7.21 The auto_make_fa_filename Routine

Purpose

This routine creates the **.fa** filename.

Input

FADT filename

Output

None

Processing

The *auto_make_fa_filename* routine performs the following task:

(1) Creates the **.fa** filename by appending ".fa" onto the FADT filename and changing "fadt" to "auto".

(2) Sets the path to the file.

Error Conditions and Handling

None

Routine Interfaces

strncpy

auto_remove_path

12.7.22 The auto_make_fa_times Routine

Purpose

This routine creates and displays the FA times for editing. The **.adv** file is used to obtain the FA times.

Input

delay_list - linked list of average delays

.fn sub-file - displayed on the user interface

Output

None

Processing

The *auto_make_fa_times* routine performs the following tasks:

(1) Calls *auto_make_fa_filename* to create a name for the **.fa** file.

(2) If the file is already displayed, brings it down.

(3) Opens the **.fa** file and displays an error message if a bad status is returned.

(4) Calls *auto_write_fa_time_line* to write the average delay times to the **.fa** file.

(5) Closes and saves the airline FA file.

(6) Displays the FA file in the rightmost pane of the report display area.

The following processing is done only when the P2E version of SCDT is being used.

(7) Loops through the entire table to get the earliest start time and latest stop time. Initializes the report start and stop times to ensure correct times are returned and converts the hour portion of the delay list to an integer. If the hour is between the report start and stop times, calls *auto_write_fa_time_line* to put the delay time in the **.adv** file.

Error Conditions and Handling

If an error occurs while the new **.fa** sub-file is being created, an error is displayed and the sub-file construction is terminated.

Routine Interfaces

strncpy

strncmp

asc_2_int

auto_open_pad

open_ios_f2_write

auto_write_flight_rec

auto_write_fa_timeline

auto_make_fa_filename

auto_display_status_message

12.7.23 The auto_make_sub_files Routine

Purpose

This routine processes the selection of the *Make Subfiles* button. Sub-files are created from the contents of the current SCDT report.

Input

reuse - boolean value

Output

.as sub-file

.edct sub-file

.al sub-file

Processing

The *auto_make_sub_files* routine performs the following tasks:

(1) Displays a status message indicating if the sub-files will be reused or new ones created.

(2) Recreates the delay list in case the FA times have been edited.

(3) Checks to see if an SCDT report is currently displayed in the report display area. If so, constructs sub-files of the report.

The following processing is done only when the P2E version of SCDT is being used.

(4) Calls *dispose_center_list* and *dispose_filght_list* to clear the center and flight lists.

Error Conditions and Handling

If the delay list cannot be re-created, sub-file processing is terminated and an error message is displayed.

Routine Interfaces

auto_open_pad

dispose_delay_list

dispose_center_list

dispose_flight_list

auto_make_al_file

auto_make_as_file

auto_make_adv_file

auto_make_edct_file

auto_read_fa_centers

reset_activation_count

auto_recreate_delay_list

auto_load_airline_addresses

add_centers_to_menu_array

auto_display_status_message

auto_load_centers_for_edct_file

auto_create_non_exempt_list

12.7.24 The auto_process_flights Routine

Purpose

This routine performs all the flight processing. This involves creating the flight lists and the sort lists (by airline, center, and delay intervals).

Input

stream handle for the current SCDT of type *ios_$id_t*.

Output

flight linked list

center linked list

airline linked list

Processing

The *auto_process_flights* routine performs the following tasks:

(1) Initializes the delay list to **nil** and calls *read_fadt_delay_list* to read the FADT delay list.

The following processing is done only when the P2E version of SCDT is being used.

(2) Initializes the flight list to **nil** and calls *create_flight_list* to create the flight list from the SCDT report.

(3) Calls *dispose_airline_list* to clear the airline list, initializes the airline list to **nil** and calls *construct_airline_list* to create a new airline list from the flight list.

(4) Initializes the center list to **nil** and calls *construct_center_list* to build a new center list from the flight list.

(5) Initializes the delay list to **nil** and calls *construct_delay_list* to build a new delay list from the flight list.

Error Conditions and Handling

None

Routine Interfaces

create_flight_list

dispose_airline_list

construct_airline_list

construct_center_list

construct_delay_list

read_fadt_delay_list

12.7.25 The auto_read_fa_centers Routine

Purpose

This routine reads each line from the **center_addresses** file and stores the data in the specified center information list.

Input

center_addresses file

Output

linked list of records - of type center_list_rec

Processing

The *auto_read_fa_centers* routine performs the following tasks:

(1) Gets the path to the **center_addresses** file.

(2) Initializes the center lists.

(3) Opens the **center_addresses** file. Displays an error message and returns when the status is bad.

(4) Reads each line 4 in the **center_addresses** file and adds it to the center list if it begins with **edct**.

(5) Closes the file.

Error Conditions and Handling

Displays an error message and stops processing the file if a bad status is returned.

Routine Interfaces

strncmp

get_word_from_buf

uto_display_status_message

12.7.26 The auto_recreate_delay_list Routine

Purpose

This routine closes the **.adv** file and reads each line into the delay list.

Input

.fa file - containing average delays

Output

fas_recreated - boolean value

delay_list - linked list with the updated average delays

Processing

The *auto_recreate_delay_list* routine performs the following tasks:

(1) Gets rid of the old delay list.

(2) Moves to the pad and closes the **.adv**. file.

(3) Opens the **.adv** file. Displays an error message, sets **auto_recreate_delay_list** to **false**, and ends processing of the file if a bad status is returned.

(4) Reads each line into the delay list.

(5) Checks the format after a possible edit. Displays an error message, closes the **.adv** file, and ends processing if an invalid format is found.

(6) Resets the delay if it is less than the cutoff value and creates or adds to the linked list.

(7) Closes the **.adv** file.

(8) Brings the airline sub-file down if it is already displayed.

Error Conditions and Handling

Displays an error message and ends processing if a bad status is returned or an invalid format is found.

Routine Interfaces

wait_for

dispose_delay_list

move_to_window

auto_display_status_message

12.7.27 The auto_subfiles_exist Routine

Purpose

This routine determines if a group of sub-files exists for the current SCDT report.

Input

.as, .edct, and .al sub-file names

Output

auto_subfiles_exist - boolean value

Processing

The *auto_subfiles_exist* routine performs the following tasks:

(1) Makes the sub-file names and gets the paths to them.

(2) Creates and displays the **.as** file

Error Conditions and Handling

None

Routine Interfaces

strncmp

auto_remove_path

auto_file_exist

12.7.28 The auto_update_fm_sequence_number Routine

Purpose

This routine updates the FM sequence number for the current node, using the specified number of FM headings in the airline sub-file.

Input

number of FA headings - integer values

Output

sequence constant - integer values

sequence number - integers values

Processing

The *auto_update_fm_sequence_number* routine performs the following tasks:

(1) Gets the path to the sequence file and opens the file. If a bad status is returned, displays an error message and attempts to create the file, giving it a default sequence number. If a bad status is returned for the create, displays an error message and does not create the file.

(2) Gets the last sequence number assigned and converts it to an integer.

(3) Calculates the new sequence number. If the last sequence number was **9999**, sets the new sequence number to **0001**.

(4) Writes the new sequence number to the file.

Error Conditions and Handling

Displays an error message and attempts to create the FA sequence file and set a default sequence number if a bad status is returned when the sequence file is opened.

Displays an error message and does *not* create the FA sequence file if a bad status is returned when the sequence file is created.

Routine Interfaces

rewind_stream

get_ios_record

auto_display_status_message

12.7.29 The auto_validate_scdt_report Routine

Purpose

This routine validates the current SCDT report, including checking the header lines. In addition, the destination airport name in the header section is saved into a global variable.

Input

stream handle for the current report - of type ios_$id_t

Output

valid file flag, indicating the report is valid - boolean value

Processing

The *auto_validate_scdt_report* routine performs the following tasks:

(1) Determines whether the fourth field of the first line in the SCDT report is the word **report**. If not, displays an error message in the *ATS* status log and ceases file validation processing.

(2) Saves the field that follows the Airport field on the second line into the **airport** global variable.

(3) If unable to find separator lines between header section and flight records, displays an error message and ceases file validation processing.

The following processing is done only when the P2E version of SCDT is being used.

(4) Determines whether the third field from the report is **analysis**; if it is, calls *auto_display_status_message* to display a message telling the user that the report is invalid.

Error Conditions and Handling

If the fourth field of the first line is *not* **report**, the file is probably an older format and is incompatible with *ATS*. An error message is placed in the *ATS* status log and validation processing is terminated.

If the separator lines between the header section and the flight records are *not* present, an error message is displayed and validation processing is terminated.

Routine Interfaces

strncmp

get_field_from_report

auto_display_status_message

12.7.30 The auto_valid_cente*r* Routine

Purpose

This routine validates the specified center (ARTCC) name by determining if the center is in a center information list.

Input

center name

name length

Output

auto_valid_center returns TRUE for a valid center name and FALSE for an invalid name

Processing

The *auto_valid_center* routine performs the following task:

When there are centers in the fa_centers_list and the center name passed in is *not* found, searches the list for that name. If the center name is found in the list, it is a valid center and the routine returns with a value of TRUE.

Routine Interfaces

strncmp

auto_get_center_address

12.7.31 The auto_write_fa_time_line Routine

Purpose

This routine writes a line of FA times to the specified stream ID, using the specified delay list.

Input

stream_id - of type ios_$id_t

delay_ptr

Output

delay_ptr - pointer to the delay linked list

Processing

The *auto_write_fa_time_line* routine writes a line of FA time to the specified **stream_id**, putting **max_fa_times_on_line** time intervals on each line.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.32 The auto_write_flight_rec Routine

Purpose

This routine writes key fields of the specified flight record to the specified stream.

Input

stream_id

flight record

Output

None

Processing

The *auto_write_flight_rec* routine performs the following tasks:

(1) Calls *strncpy* to copy the **flight ID**, the **departure center**, and the **ptime** from the flight record to the data record.

(2) Calls *cal_delay* to calculate the **delay** for the flight and calls *strncpy* to copy the **delay** to the data record. Uses the **ptime** if the **delay** is *less* than the cutoff time, and uses the **ctime** if the **delay** is *greater* than the cutoff time.

(3) Calls *strncpy* to copy the **cta** to the data records.

(4) Writes the data record to the indicated **stream_id**.

Error Conditions and Handling

None

Routine Interfaces

strncpy

cal_delay

12.7.33 The auto_write_preface_list Routine

Purpose

This routine writes the contents of the preface list to the specified stream identification.

Input

preface_list

stream_id

Output

None

Processing

The *auto_write_preface_list* routine performs the following tasks:

(1) Loads **preface_list** into a temporary list, **place**.

(2) When the list is *not* empty, writes the contents of the list to **stream_id**.

Error Conditions and Handling

None

Routine Interfaces

None

12.7.34 The cal_delay Routine

Purpose

This routine calculates flight delay intervals.

Input

ptime

ptime_len

edct

edct_len

Output

delay interval

Processing

The *cal_delay* routine performs the following tasks:

(1) Stores **ptime** and **etime** for possible changes.

(2) If a time is passed in with a prefix, strips off the prefix.

(3) Converts the times to integers. To avoid time changes, sets the delay to the maximum and returns when the status is bad.

(4) Gets the minutes and the hour of the **ptime** and **etime**.

(5) If the **etime** is *less* than the **ptime**, assumes midnight crossover and adds **24** to the **etime** hour.

(6) Converts the total times to minutes and calculates the delay interval.

Error Conditions and Handling

If an error is encountered when converting the times to integers, sets the delay to the maximum and returns to avoid time changes.

Routine Interfaces

None

12.7.35 The load_international_centers Routine

Purpose

This routine loads the international centers list.

Input

international_centers file

Output

linked list of international centers

Processing

The *load_international_centers* routine performs the following tasks:

(1) Finds and opens the **international_centers** file. Displays an error message and ends processing if the file cannot be opened.

(2) Resets the **international_centers** list to NULL.

(3) Reads the file, searches for the center name, and adds it to the list.

Error Conditions and Handling

Displays an error message and ends processing if a bad status is returned while opening the **international_centers** file.

Routine Interfaces

get_word_from_buf

12.7.36 The read_fadt_delay_list Routine

Purpose

This routine reads the FADT delay list to get the delay data.

Input

FADT file

Output

linked list of delay times

Processing

The *real_fadt_delay_list* routine performs the following tasks:

(1) Opens the current FADT report. Displays an error message and returns when the status is bad.

(2) Searches the FADT report for delays. Displays an error message, closes the FADT report, and returns if no delays are found.

(3) Calls *get_word_from_buf* to find **delay** and **average** in the FADT report.

(4) Calls *dispose_delay_list* to get rid of the delay list and sets it to **nil**.

(5) Resets the delay list.

(6) Gets the header line.

(7) Reads through the FADT report and gets the delay intervals until the B6 section is found, a bad status is returned, or **word_len** is *not* **4**. If a bad status is encountered, displays an error message telling the user that there were problems reading the delay list in the FADT report.

(8) Closes the FADT report.

Error Conditions and Handling

Displays an error message when a bad status is returned.

Routine Interfaces

strncmp

get_word_from_buf

dispose_delay_list

auto_display_status_message

12.8 The ats_send Module

Purpose

This module contains routines, which deal with sending *ATS* files and information. Figure 12-4 shows the data flow of the *ats_send* module.

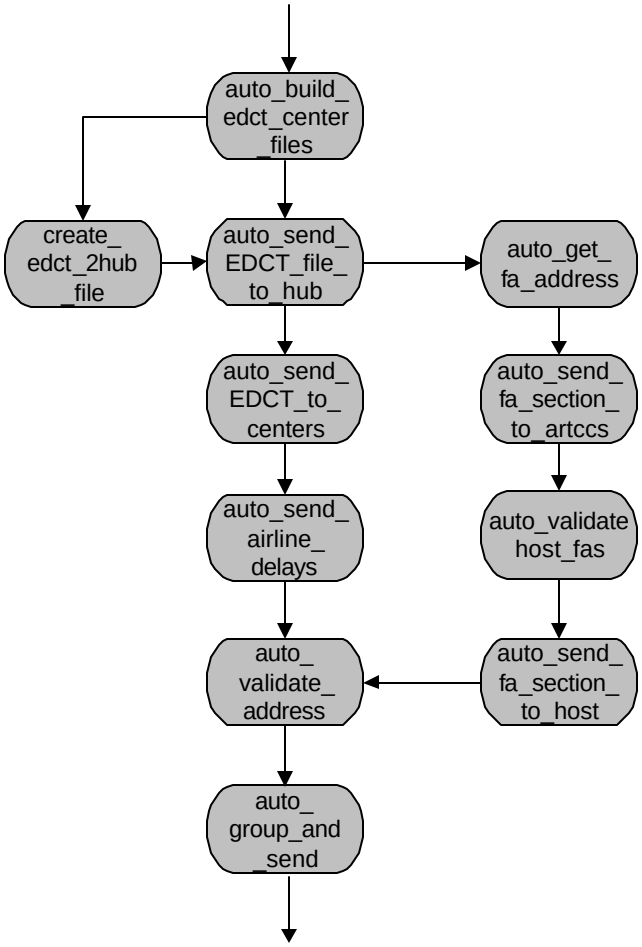

Figure 12-4. Data Flow of the *ats_send* Module

12.8.1 The auto_build_edct_center_files Routine

Purpose

This routine builds the EDCT files for the centers.

Input

menu array - array of center names

choices - selected menu choices

number of choices

Output

built - boolean value

Processing

The *auto_build_edct_center_files* routine performs the following tasks:

(1) If **built** is false, calls *delete_exxx_files* to delete all EDCT files.

(2) Opens the EDCT file and returns when status is bad.

(3) Reads the EDCT file until the **fad** line is found and gets the destination airport.

(4) Reads the EDCT file until the **dep** line is found and gets the center name. If the selection is the first choice, sets **found** to **true**. If it is not, searches for the selection in the menu array and sets **found** to **true** when it is found.

(5) If the selection is found in the menu array, creates a name for the send file for the indicated center and opens a stream to the send file. Displays an error message and goes to the next center if a bad status is returned.

(6) Writes **edct airport** to the send file.

(7) Writes the **dep** line out to the send file. Quits when a blank line is encountered.

(8) Closes the send file.

Error Conditions and Handling

Displays an error message and returns when status is bad.

Routine Interfaces

get_word_from_buf

auto_display_status_message

strncmp

delete_exxx_files

12.8.2 The auto_do_keyword_select Routine

Purpose

This routine sends the contents of a file to the addresses in the specified keyword file.

Input

addresses from the address keyword file

file name to be sent to the *RTR* process

keyword file identifier

Output

file sent to *RTR*

Processing

The *auto_do_keyword_select* routine performs the following tasks:

(1) Constructs a complete keyword file pathname using the specified keyword identifier.

(2) Opens the keyword file for reading. If the keyword file cannot be opened, displays an error message and ceases keyword processing.

(3) Performs the following actions for each address type in the keyword file:

  (a) Finds the start of an address type by searching for specific identifiers.

  (b) Once the start of an address type is found, loads all the following addresses into a temporary address list until the start of a new address type is reached. Validates each address as it is placed in the list, skipping the address if it is invalid in syntax.

  (c) Fills in fields of the mailbox record, including the name of the file to be sent and the current date and time.

  (d) Depending on the type of addresses, fills in the **type**, **priority**, and **destination** fields of the *RTR* mailbox record.

(e) If the keyword identifier was **adv** (for advisory files), sets the **type** field of the mailbox record to indicate that the file to be sent is an advisory file.

(f) Sends the mailbox record to the *RTR* process using the temporary address list for addressing.

(g) Disposes of the temporary address list.

(4) Closes the address keyword file.

Error Conditions and Handling

If the specified address keyword file cannot be opened for reading, an error message is placed in the *ATS* status log and keyword processing is terminated.

If an address read from the address keyword file is invalid in syntax, it is skipped and keyword processing continues with the next address.

Routine Interfaces

strncmp

strncpy

get_word_from_buf

auto_group_and_send

auto_dispose_temp_list

auto_validate_address

auto_display_status_message

12.8.3 The auto_get_fa_address Routine

Purpose

This routine reads the destination addresses for the FA sections of the airline sub-file from an address file and stores them in a temporary address list.

Input

addresses from the fa_addresses file

FA section data from the airline sub-file

Output

FA section information written to a send file

Status messages

Error messages

Processing

The *auto_get_fa_address* routine performs the following tasks:

(1) Initializes local variables.

(2) Gets a line from the FA address file.

(3) Calls *get_word_from_buf* to get a word from the line.

(4) Checks that **word_len** is not **0** and calls *auto_validate_address* to check that the address is valid. If the address is valid, adds it to the list. If the address is *not* valid, displays an error message to the specialist.

(5) Disposes of the temporary address list.

(6) Redisplays the airline sub-file in the report display area.

Error Conditions and Handling

If an invalid address is found, an error message is displayed to the specialist.

Routine Interfaces

strncpy

get_word_from_buf

uto_display_status_message

auto_validate_address

12.8.4 The auto_group_and_send Routine

Purpose

This routine sends a global mailbox record to the *RTR* process, using a specified list of addresses. (Multiple copies of the mailbox record are sent-the maximum number of addresses at a time.)

Input

address list

Output

error message

mailbox record sent to the *RTR* process

Processing

The *auto_group_and_send* routine performs the following tasks:

(1) Depending on the address types (determined by checking the global mailbox record), determines the maximum number of addresses that can be included in the mailbox record. If the address type contained in the global mailbox record is invalid, displays an error message and ceases sending.

(2) Traversing through the address list, loads the maximum number of addresses possible into the mailbox record and sends it.

(3) Continues loading groups of addresses and sending the mailbox record until the list of addresses has been completely traversed.

Error Conditions and Handling

If the address type located in the global mailbox record is invalid, an error message is placed in the *ATS* status log and sending is terminated.

Routine Interfaces

strncpy

auto_display_status_message

12.8.5 The auto_send_airline_delays Routine

Purpose

This routine sends airline delays to all airlines selected from the airline menu.

Input

array of selected airlines from the airline menu

number of choices

Output

None

Processing

The *auto_send_airlines_delays* routine performs the following tasks:

(1) Opens the **.al** file for a read. Displays an error message and returns when status is bad.

(2) Sets up the message header.

(3) Searches the menu for selected airlines.

(4) When a selected airline is found, calls *auto_get_airline_addresses* to get the address for the airline and finds the first section for this airline.

(5) Reads records from the airline file until the selected airline is found. If an airline is not found, adds it to the missing airlines array, increments **index**, and exits.

(6) Calls *auto_create_section_file_name* to get the section file name and creates the file. Closes the stream, displays an error message, and returns when status is bad.

(7) Puts the flight data into the section file until end of file or a new section is encountered. Exits at end of file. If at a new section of the file, determines if it is for the same airline. Exits if it is *not* the same airline. If it is the same airline, closes the current section file and creates a new file for the new section. If *not* at a new section, puts the data into the file.

(8) Closes the open streams.

(9) Copies the section file to **send_msg** and sends the file.

(10) Disposes of the temporary list.

(11) Closes the open streams.

(12) Displays an error message saying that delays have not been sent to indicated airlines if the sections could not be found.

Error Conditions and Handling

If there are airlines in the airline selection menu, an error message is placed in the *ATS* status log and airline selection processing is terminated.

If no airlines have been selected, an error message is placed in the *ATS* status log and airline selection processing is terminated.

If any error occurs during the sending of airline information, an error message is displayed in the *ATS* status log.

Routine Interfaces

strncpy

strncmp

auto_display_status_message

rewind_stream

get_word_from_buf

uto_group_and_send

auto_dispose_temp_list

auto_get_airline_addresses

auto_create_section_file_name

12.8.6 The auto_send_edct_file_to_hub Routine

Purpose

This routine sends the entire EDCT file to the hubsite to update internal flight tables

Input

EDCT file

Output

msg

msg_len

Processing

The *auto_send_edct_file_to_hub* routine performs the following tasks:

(1) Gets the choices from the EDCT menu and checks the status. Returns when status is bad.

(2) If there are no airlines in the airline selection menu, displays an error message and ceases processing of selected airlines.

(3) If no airlines were selected, displays an error message and ceases processing of selected airlines.

(4) Closes the airline sub-file and the open window.

(5) Sends the airline delay information to all selected airlines.

(6) If any errors occurred while sending airline information, displays an error message. Otherwise, displays a successful send status message.

(7) Calls *auto_reset_menu* to reset the *EDCT* menu.

(8) Sets up the send header

(9) Calls *auto_send_it* to send the file.

Error Conditions and Handling

If there are airlines in the airline selection menu, an error message is placed in the *ATS* status log and airline selection processing is terminated.

If no airlines have been selected, an error message is placed in the *ATS* status log and airline selection processing is terminated.

If any error occurs during the sending of airline information, an error message is displayed in the *ATS* status log.

Routine Interfaces

strncpy

strncmp

auto_display_status_message

rewind_stream

get_word_from_buf

uto_group_and_send

auto_dispose_temp_list

auto_reset_menu

auto_send_it

12.8.7 The auto_send_edct_to_centers Routine

Purpose

This routine sends the sub-edct file to the centers.

Input

None

Output

None

Processing

The *auto_send_edct_to_centers* routine performs the following tasks:

(1) Initializes local variables.

(2) Gets the chosen centers from the *EDCT Centers* menu.

(3) Builds a file for each center chosen from the menu, including only the flights for that center; then calls *build_auto_footer* to add the footer to the file and *auto_do_keyword_select* to send the file to the center. If a bad status is encountered, calls *auto_display_status_message* to display an error.

(4) Calls *auto_reset_menu* to clear the menu choices.

Error Conditions and Handling

If an invalid address is found, an error message is displayed to the specialist.

Routine Interfaces

auto_reset_menu

build_auto_footer

auto_do_keyword_select

auto_display_status_message

12.8.8 The auto_send_fa_section_to_artccs Routine

Purpose

This routine sends individual FA sections for each ARTCC.

Input

selection array

number of choices

Output

FA section information written to a send file

Processing

The *auto_send_fa_section_to_artccs* routine performs the following tasks:

(1) Opens the **.as** file for a read. Displays an error message and returns when status is bad.

(2) Calls *auto_create_section_file_name* to get the section file name and creates the file. Closes the stream, displays an error message, and returns when status is bad.

(3) Puts the flight data into the section file until the FM section, the airline section, or the end of file is encountered. Exits at end of file or at the next section of the **.as** file. Skips the line if it is an FA section header.

(4) If an FA section is found for the center, adds the footer to the end of the file. If *no* section is found, displays an error message, closes the open streams and returns.

(5) Closes the open streams.

(6) Loads the menu selections and builds the addresses. Compares the FA menu selection to the mapped FA menu array to validate the address. If the address is valid, it is added to the address list. If the address is invalid, it is added to the bad center list.

(7) Closes the open streams.

(8) If the FA section is found and the address list is *not* empty, sets up the message header, sends the FA section to the centers in the address list, and displays a status message that the FAs have been sent.

(9) Displays an error message that FAs have not been sent to indicated centers if the center address is invalid.

(10) Disposes of the temporary address list.

Error Conditions and Handling

If an invalid address is found, an error message is displayed to the specialist.

If the status is bad, an error message is displayed and the routine returns.

Routine Interfaces

strncmp

strncpy

rewind_stream

read_rec_from_file

get_word_from_buf

uto_validate_address

auto_validate_host_fas

auto_dispose_temp_list

auto_display_status_message

auto_create_section_file_name

12.8.9 The auto_send_fa_section_to_host Routine

Purpose

This routine sends individual FA sections for each center.

Input

selection array

number of choices

Output

FA section information written to a send file

Processing

The *auto_send_fa_section_to_host* routine performs the following tasks:

(1) Opens the **.as** file for a read. Displays an error message and returns when status is bad.

(2) Displays a message that the Host FA section is being validated.

(3) Calls *auto_validate_host_fas* to validate the FA section to be sent. Closes the open stream, displays an error message, and returns if the FA section is invalid.

(4) Calls *auto_create_section_file_name* to get the section file name and creates the file. Closes the stream, displays an error message, and returns when status is bad.

(5) Searches the FA menu for selected centers. When a selected center is found, finds its first FA section and reads the **.as** file for all sections for that center. If a selected center does not have a section, adds the center to the missing centers list, increments **index**, and exits. Searches for the next selected center if there are no more sections.

(6) If there are sections for the selected center, reads the FA address file, calls *auto_validate_address* to validate the address, and adds it to the list if it is

good. If the address is invalid, calls *beep* to signify an error, and displays error messages to the specialist.

(7) If the address is for the indicated center and it is valid, adds it to the FA section file.

(8) Closes the open streams.

(9) If the FA section is found, sets up the message header, sends the FA section to the centers in the address list, and displays a status message saying that the FAs have been sent.

(10) If FA sections were not found for the centers, displays an error message saying that FAs have not been sent to the indicated centers.

(11) Disposes of the temporary address list.

Error Conditions and Handling

If an invalid address is found, an error message is displayed to the specialist.

If the status is bad from the open file call, a message is displayed and the routine returns.

Routine Interfaces

strncmp

strncpy

rewind_stream

read_rec_from_file

get_word_from_buf

uto_validate_address

auto_validate_host_fas

auto_dispose_temp_list

auto_display_status_message

auto_create_section_file_name

12.8.10 The auto_send_fa_section_to_extra_fa_addr Routine

Purpose

This routine sends the FA section of the airline sub-file to an extra address listed in the **fa_addresses** file.

Input

.as file

Output

FA times sent to the addresses adapted in the **setup** file

Processing

The *auto_send_fa_section_to_extra_fa_addr* routine performs the following tasks:

(1) Opens the **.as** file and checks the status. Displays an error message and exits if a bad status is returned.

(2) Calls *auto_create_section_file_name* and creates the section file. Closes the streams, displays an error message, and returns when status is bad.

(3) Puts the flight data into the section file until the FM section, the airline section or the end of file is found. Closes the open streams.

(4) If an FA section is not found, displays an error message and returns.

(5) Sets a temporary linked list to the extra FA address linked list.

(6) If an FA section is found and there are addresses in the list, sets up the message header and calls *auto_group_and_send* to send the FA section.

(7) Displays an error message if an FA section is not found or there are no addresses in the list.

Error Conditions and Handling

Displays an error message and returns if a bad status is returned.

Displays an error message if an FA section is not found or the address list is empty.

Routine Interfaces

strncpy
rewind_stream
auto_group_and_send
auto_display_status_message
auto_create_section_file_name

12.8.11 The auto_send_it Routine

Purpose

This routine places the current mailbox record in the *RTR* process mailbox channel.

Processing

The *auto_send_it* routine performs the following tasks:

(1) Initializes local variable **st_msg** to **no_msg**.

(2) Gets the full pathname of the file to be sent and checks the status.

(3) If the status is good, calls *transfer_msg_for_send* to send the file

(4) If the status is *not* good, calls *vfmt_$encode* to encode an error message and calls *auto_display_status_message* to display the error to the specialist.

Error Conditions and Handling

If the file is not found, an error message is displayed.

Routine Interfaces

strncmp

transfer_msg_for_send

auto_display_status_message

12.8.12 The auto_validate_address Routine

Purpose

This routine checks the syntax of the specified address, depending on the specified address type.

Input

dest - of type dest_type_t

Output

bad_address - boolean value

Processing

The *auto_validate_address* routine performs the following tasks:

(1) Sets **bad_address** to **false**.

(2) Determines if the address is a valid ETMS address. If address length is between **5** and **9** and the destination is **etms**, the address is valid. If the address is invalid and it is not in the **valid_nadin_arinc_chars** file, sets **bad_address** to **true** and returns.

(3) If the destination is **arinc**, sets **bad_address** to **true** if **address_len** is *not* **7**.

(4) If the destination is **nadin**, sets **bad_address** to **true** if **address_len** is *not* **6** or **8**.

(5) If the address is a valid ETMS address with a length between **4** and **42**, validates the address.

(6) If the first character of the address is **$**, encodes **.all** into the address.

(7) If the address contains a *wildcard*, determines that it is **all**, **any**, or **this**. If the wildcard is not any of these, sets **bad_address** to **true**.

Error Conditions and Handling

If an invalid format is found, **bad_address** is set to **true**.

Routine Interfaces

strncmp

12.8.13 The auto_validate_host_fas Routine

Purpose

This routine checks the syntax of the specified host FA section.

Input

stream - of type ios_$id_t

Output

error messages - character string

Processing

The *auto_validate_host_fas* routine performs the following tasks:

(1) Initializes local variables.

(2) Reads the Host FA section passed in. Returns if an FA section is not found.

(3) Checks the address. If a bad address is encountered, encodes an error message and exits.

(4) Advances to the next line.

(5) Checks for a sequence number. If none is found, encodes an error message and exits.

(6) Advances to the next line.

(7) Sets **pointer** to **8**. Checks if there is anything after the pointer. If there is text after the pointer, validates FA line parameters. If there is no text after the pointer or the FA line parameters are invalid, encodes an error and exits.

(8) Advances to the next line.

(9) Checks that this is a blank line. If it is not, encodes an error and exits.

(10) Resets **addrs**, **dcc**, **blank**, and **fa_buf** and reads the next FA section.

Error Conditions and Handling

If an invalid line format is found, displays an error message and exits.

Routine Interfaces

None

12.8.14 The cat_center_edcts_file Routine

Purpose

This routine creates EDCT files to be sent to the hub.

Input

to_file_name

to_file_len

from_file_name

from_file_len

Output

status

Processing

The *cat_center_edcts_file* routine performs the following tasks:

(1) Opens a stream to **from_file_name.** If a bad status is encountered, returns control to the calling procedure.

(2) Finds the length of the buffer. If a bad status is encountered or the buffer is empty, closes the stream to **from_file_name**, sets the status to **-1**, and returns control to the calling procedure.

(3) Opens a stream to **to_file_name.** If a bad status is encountered, closes the stream and returns control to the calling procedure.

(4) Reads each line in the file, starting from the second line of text in **from_file_name**. If the status is good and **read_buf_len** is greater than **0**, increments **accum_buf_len** by **read_buf_len** and puts the contents of the buffer into **from_file_name**.

(5) Closes the open streams.

Error Conditions and Handling

If a bad status is encountered when opening a file, closes the stream to that file and returns control to the calling procedure.

Routine Interfaces

None

12.8.15 The create_edct_2hub_file Routine

Purpose

This routine creates EDCT files to be sent to the hub.

Input

to_file_name

to_file_len

Output

status

Processing

The *create_edct_2hub_file* routine performs the following tasks:

(1) Calls *ios_$create* to create the file to be sent to the hub and opens a stream to the file. If a bad status is encountered, returns control to the calling procedure.

(2) Calls *ios_$open* to open a stream to the EDCT file. If a bad status is encountered, closes the stream to the EDCT file and returns control to the calling procedure.

(3) Reads each line of the EDCT file.

(4) Gets each word on the line and puts it into **str**. Stops if a bad status is encountered or **fadt** found.

(5) If the status is good, puts the contents of the EDCT file into the new file.

(6) Closes the open streams.

Error Conditions and Handling

If a bad status is encountered when creating the file, returns control to calling procedure.

If a bad status is encountered reading the EDC*T* file or **fadt** is found, processing stops.

If a bad status is encountered when opening the EDCT file, closes the stream to **to_file_name** and returns control to the calling procedure.

Routine Interfaces

get_word_from_buf

ios_$create

ios_$open

strncmp

12.8.16 The delete_exxx_files Routine

Purpose

This routine deletes the EDCT files.

Input

None

Output

None

Processing

The *delete_exxx_files* routine performs the following tasks:

(1) Initializes local variables.

(2) Calls *vfmt_$encode* to encode the command to delete all EDCT files.

(3) Sets up **connv** and **argvector** and calls *pgm_$invoke* to execute the delete command.

Error Conditions and Handling

None

Routine Interfaces

vfmt_$encode

pgm_$invoke

12.9 ATS Data Structure Tables

Tables 12-1 through 12-14 describe data structures used by *ATS***.** Descriptions of data structures used internally by *ATS* are included as well as those used for communicating between this and other processes. The tables describe each data item within the data structures. The description includes a definition, unit/format, range, and variable type. When the unit/format field or range field is blank, it indicates that a specific unit/format or range was not defined.

12.9.1 The delay_rec Data Structure

The **delay_rec** data structure is used by the *ATS* process to store a list of delay intervals.

Table 12-1. delay_rec Data Structure

delay_rec				
Library Name: auto_send_lib		**Purpose:** Stores a list of delay intervals		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
start	Delay interval start time.			field_type
stop	Delay interval stop time.			field_type
count	Number of delays.			integer
total	Total time of delays.			integer
average	Average time of delays.			integer
next	Pointer to sublist.			delay_list_type

12.9.2 *The* set_record_t Data Structure

The **set_record_t** data structure is used by the *ATS* process to extract data from the user interface for a particular airport or fix.

Table 12-2. set_record_t Data Structure

set_record_t				
Library Name: auto_send_lib		**Purpose:** Storing airport and fix data		
Element Name: ats_cont_p.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
name	Airport/fix name.	2-12 characters		data_t
start_time	Start time.	HHMM	0000 - 2359	data_t
stop_time	Stop time.	HHMM	0000 - 2359	data_t
start_date	Start date.	MMDD		data_t
ete_q	Estimated time enroute queue.			data_t
ete	Estimated time enroute.	minutes	0 - 999	data_t
airp_accept_rate	Airport acceptance rate.	aircraft/hour	1 - 999	data_t
accept_rate	Acceptance rate.			data_t
ga_factor	General aviation factor.	aircraft/hour	9 - 999	data_t
stack	Stack size.	number of aircraft holding	0 - 999	data_t
minimum_delay	Minimum delay.	minutes	0 - 999	data_t
maximum_delay	Maximum delay.	minutes	0 - 999	data_t
airborne_delay	Airborne delay.	minutes	0 - 999	data_t
dept_airp	Departure airport.	2-5 characters	1 to max_cntr_airp	data_t
dept_cntr	Departure center.	3 characters	1 to max_cntr_airp	data_t
aircraft_type	Aircraft type.	2-4 characters	1 to max_atype	data_t
aircraft_class	Aircraft class.	1-8 characters	1 to max_class	data_t
aircraft_category	Aircraft category.	character	O, T, F, C, G, M	data_t
dept_airp_adj	Departure airport adjustment.	2-5 characters	1 to max_adj	data_t
dept_cntr_adj	Departure center adjustment.	3-4 characters	1 to max_adj	data_t

dept_airp_q	Departure airport queue.			data_t

Table 12-3. Table 12-2. set_record_t Data Structure (continued)

set_record_t				
Library Name: auto_send_lib		**Purpose:** Data record for a particular airport or fix		
Element Name: ats_cont_p.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
dept_cntr_q	Departure center queue.			data_t
aircraft_type_q	Aircraft type queue.			data_t
aircraft_class_q	Aircraft class queue.			data_t
aircraft_category_q	Aircraft category queue.			data_t
user_category_q	User category queue.			data_t
num_dept_airp	Number of departure airports			integer
num_dept_cntr	Number of departure centers		1 – 20	integer
num_aircraft_type	Number of aircraft types.			integer
num_aircraft_class	Number of aircraft class.			integer
num_aircraft_category	Number of aircraft category.			integer
num_user_category	Number of user category.			integer
num_dept_airp_adj	Number departure airport adj.		1 – 10	integer
num_dept_cntr_adj	Number departure center ady.		1 – 10	integer

12.9.3 The key_record Data Structure

The **key_record** data structure is used to read in data to initialize the **key_tables** used during SCDT report processing.

Table 12-4. key_record Data Structure

key_record				
Library Name: auto_send_lib		**Purpose:** Record type for search key		
Element Name: ats_cont_p.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
keyt	Search key.			string
key_len	Key length.			integer

12.9.4 The pad_record_t Data Structure

The **pad_record_t** data structure is used by the *ATS* process to store information about the window pads.

Table 12-5. pad_record_t Data Structure

pad_record_t				
Library Name: auto_send_lib		**Purpose:** Stores pad information		
Element Name: ats_auto_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
desc	Description of pad window.			pad_$window_desc_t
stream	Stream.			ios_$id_t
file_name	Name of file.			string
file_name_len	Length fo file			integer

12.9.5 The temp_rec Data Structure

The **temp_rec** data structure is used by the *ATS* process as a temporary storage area during local routine processing.

Table 12-6. temp_rec Data Structure

temp_rec				
Library Name: auto_send_lib		**Purpose:** Provides a temporary storage area during local routine processing		
Element Name: ats_auto_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
data	Data			data_t
data_len	Length of data.			integer
next	Next list entry.			temp_list_type

12.9.6 The flight_record_t Data Structure

The **flight_record_t** data structure is used to store flight information

Table 12-7. flight_record_t Data Structure

flight_record_t				
Library Name: auto_send_lib		**Purpose:** Stores flight information		
Element Name: ats_auto_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
ident	Flight id			char20_t
ident_len	Length of flight id field			integer
dept	Departure center			char20_t
dept_len	Length of departure center field			integer
ptime	Proposed departure time			char20_t
ptime_len	Length of ptime field			integer
ctime	Control time			char20_t

ctime_len	Length of ctime field			integer
cta	Control time of arrival			char20_t
cta_len	Length of control time field			integer

12.9.7 The flight_rec Data Structure

The **flight_rec** data structure is used by the *ATS* process to store flight information.

Table 12-8. flight_rec Data Structure

flight_rec				
Library Name: auto_send_lib		**Purpose:** Stores flight information		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
ident	Flight id			field_type
ident_len	Length of flight id field			integer
dept	Departure center			field_type
dept_len	Length of departure center field			integer
center	Center			field_type
center_len	Length of center field			integer
ptime	Proposed departure time			field_type
ptime_len	Length of ptime field			integer
ctime	Control time			field_type
ctime_len	Length of ctime field			integer
eta	Estimated time of arrival			integer
cta	Control time of arrival			field_type
cta_len	Length of cta field			integer
delay	Delay interval			integer
active	Active flight determination			boolean
exempt	Exempt flight determination			boolean

ptime_prefix	Prefix before ptime			char

12.9.8 The flight_list_rec Data Structure

The **flight_list_rec** data structure stores a list of flights.

Table 12-9. flight_list_rec Data Structure

flight_list_rec				
Library Name: auto_send_lib		**Purpose:** Stores a list of flights		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
flight	Flight id			flight_rec
next	Pointer to sublist			flight_list_type

12.9.9 The ptr_list_rec Data Structure

The **ptr_list_rec** data structure stores a list of pointers.

Table 12-10. ptr_list_rec Data Structure

ptr_list_rec				
Library Name: auto_send_lib		**Purpose:** Stores a list of pointers		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
flight_ptr	Pointer to flight			flight_rec
next	Pointer to sublist			flight_list_type

12.9.10 The air_list_rec Data Structure

The **air_list_rec** data structure is used to store a list of airlines.

Table 12-11. air_list_rec Data Structure

air_list_rec				
Library Name: auto_send_lib		**Purpose:** Stores a list of airlines		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
name	Airline name			field_type
name_len	Length of name field			integer
count	Number of flights			integer
delay	Flag indicating delays present			boolean
flights	Pointer to a list of flights			ptr_list_type
next	Pointer to a sublist			air_list_type

12.9.11 The center_list_rec Data Structure

The **center_list_rec** data structure is used to store a list of centers.

Table 12-12. center_list_rec Data Structure

center_list_rec				
Library Name: auto_send_lib		**Purpose:** Stores a list of centers		
Element Name: ats_list_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
name	Center name			field_type
name_len	Length of name field			integer
count	Number of flights			integer
flights	Pointer to a list of flights			ptr_list_type
next	Pointer to a sublist			center_list_type

12.9.12 The file_list_record_t Data Structure

The **file_list_record_t** data structure is a general data type used to store file names and the length of file names.

Table 12-13. file_list_record_t Data Structure

file_list_record_t				
Library Name: auto_send_lib		**Purpose:** Stores file names and the length of file names.		
Element Name: ats_auto_types.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
name	File name.			string
name_len	Name length.			integer

12.9.13 The data_t Data Structure

The **data_t** data structure is a general data type used for array processing in the *ATS* process.

Table 12-14. data_t Data Structure

data_t				
Library Name: auto_send_lib		**Purpose:** Data record type for data extracted for SCDT reports.		
Element Name: ats_cont_p.ins.pas				
Data Item	**Definition**	**Unit/Format**	**Range**	**Var. Type/Bits**
str	Data array.	1 – 20 characters		array of char
str_len	String length.			integer

12.10 *Autosend* Cross Reference List

The following list can be used to locate routines within the *Autosend* section.